#3
LTyson
03-22-01

Docket No.: 49657-921

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of

Toyohiko YOSHIDA

Serial No.:

Group Art Unit:

Filed: January 10, 2001

Examiner:

For: INSTRUCTION TRANSLATOR TRANSLATING NON-NATIVE INSTRUCTIONS FOR A PROCESSOR INTO NATIVE INSTRUCTIONS THEREFOR, INSTRUCTION MEMORY WITH SUCH TRANSLATOR, AND DATA PROCESSING APPARATUS USING THEM

CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Commissioner for Patents
Washington, DC 20231

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claims the priority of:

Japanese Patent Application No. 2000-005062,
filed January 13, 2000

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:klm
Date: January 10, 2001
Facsimile: (202) 756-8087

49657-921
JANUARY 10, 2001
YOSHIDA
McDermott, Will & Emery

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２０００年 １月１３日

出 願 番 号
Application Number: 特願２０００－００５０６２

出 願 人
Applicant (s): 三菱電機株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

２０００年 ２月 ４日

特許庁長官
Commissioner,
Patent Office 近藤隆彦

出証番号 出証特２０００－３００４１６３

【書類名】　特許願

【整理番号】　520274JP01

【提出日】　平成12年 1月13日

【あて先】　特許庁長官殿

【国際特許分類】　G06F　9/30

G06F　9/455

【発明者】

【住所又は居所】　東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

【氏名】　吉田　豊彦

【特許出願人】

【識別番号】　000006013

【氏名又は名称】　三菱電機株式会社

【代理人】

【識別番号】　100064746

【弁理士】

【氏名又は名称】　深見　久郎

【選任した代理人】

【識別番号】　100085132

【弁理士】

【氏名又は名称】　森田　俊雄

【選任した代理人】

【識別番号】　100091409

【弁理士】

【氏名又は名称】　伊藤　英彦

【選任した代理人】

【識別番号】　100096781

【弁理士】

【氏名又は名称】　堀井　豊

【選任した代理人】

【識別番号】　100096792

【弁理士】

【氏名又は名称】　森下　八郎

【手数料の表示】

【予納台帳番号】　008693

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　命令トランスレータ、トランスレータ付命令メモリおよびそれらを用いたデータ処理装置

【特許請求の範囲】

【請求項１】　第１の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、前記第１の命令体系と異なる第２の命令体系の命令を記憶する命令メモリとともに用いられ、前記第１の命令体系の命令に変換して前記プロセッサに与えるための命令トランスレータであって、

前記プロセッサが実行すべき命令の、前記命令メモリにおけるアドレスを受けて前記命令メモリから対応する命令を読出し、読出された前記第２の命令体系の命令を前記第１の命令体系の命令に変換するための変換手段と、

前記変換手段により変換された後の前記第１の命令体系の命令を、前記命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、

前記プロセッサが実行すべき命令のアドレスを受けて前記保持手段を探索し、前記保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、前記変換手段の出力する命令と、前記保持手段に保持されていた、対応の前記第１の命令体系の命令とを選択的に前記プロセッサに出力するための選択手段とを含む、命令トランスレータ。

【請求項２】　前記第２の命令体系は可変長命令体系であり、前記変換手段は、前記命令メモリから読出された前記第２の命令体系の命令を、前記読出された前記第２の命令体系の命令の命令長に依存した数の前記第１の命令体系の命令に変換するための可変長命令変換手段を含む、請求項１に記載の命令トランスレータ。

【請求項３】　前記可変長命令変換手段は、前記命令メモリから読出された前記第２の命令体系の命令を、前記読出された前記第２の命令体系の命令の命令長に依存し、かつそれよりも長い長さの前記第１の命令体系の命令に変換するための第１の手段を含む、請求項２に記載の命令トランスレータ。

【請求項４】　前記第１の命令体系の各命令は１または複数個のサブ命令を含み、前記第１の手段が変換する第１の命令体系の命令に含まれるサブ命令の個

数は、前記読み出された前記第２の命令体系の命令の長さに依存する、請求項３に記載の命令トランスレータ。

【請求項５】　前記変換手段は、前記命令メモリから読出された前記第２の命令体系の複数個の命令を、１つの前記第１の命令体系の命令に変換するための複数命令変換手段を含む、請求項１に記載の命令トランスレータ。

【請求項６】　前記第１の命令体系の各命令は、１または複数個のサブ命令を含むことが可能であり、

前記変換手段は、前記命令メモリから読出された前記第２の命令体系の複数個の命令を、前記複数個の命令の数に依存した数のサブ命令を含む、前記第１の命令体系の命令に変換するための手段を含む、請求項１に記載の命令トランスレータ。

【請求項７】　前記変換後の前記第１の命令体系の命令に含まれるサブ命令の個数が、前記複数個の命令の数と等しい、請求項６に記載の命令トランスレータ。

【請求項８】　第１の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、前記第１の命令体系と異なる第２の命令体系の命令を記憶する命令メモリとともに用いられ、前記第１の命令体系の命令に変換して前記プロセッサに与えるための命令トランスレータであって、

前記プロセッサが実行すべき命令の、前記命令メモリにおけるアドレスを受けて前記命令メモリから対応する命令を読出し、読出された前記第２の命令体系の命令を前記第１の命令体系の１または複数個の命令に変換する変換手段と、

前記変換手段により変換された後の前記第１の命令体系の命令を、前記命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、

前記プロセッサが実行すべき命令のアドレスを受けて前記保持手段を探索し、前記保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、前記変換手段の出力する命令と、前記保持手段に保持されていた、対応の前記第１の命令体系の命令とを選択的に前記プロセッサに出力するための選択手段と、

前記保持手段に保持されている命令を第１の条件および第２の条件のいずれか

で無効化可能なエントリとして保持するように前記保持手段を制御するための保持制御手段とを含む、命令トランスレータ。

【請求項９】　前記第１の条件は前記保持手段による所定のアルゴリズムに基づくハードウェア制御による保持制御の条件であり、

前記第２の条件は前記保持手段からの外部からの明示的な無効化指示があったという条件である、請求項８に記載の命令トランスレータ。

【請求項１０】　前記保持制御手段は前記第２の条件で無効化が可能なエントリを無効化することなしに新たな命令を前記保持手段に保持できないときにアサートされる信号を出力する、請求項８に記載の命令トランスレータ。

【請求項１１】　第１の命令体系の命令をネイティブ命令として動作するプロセッサとともに用いられる命令メモリであって、

第２の命令体系の命令を記憶する命令記憶手段と、

前記命令記憶手段から出力される前記第２の命令体系の命令を、前記第１の命令体系の命令に変換して前記プロセッサに与えるための命令トランスレータとを含む、トランスレータ付命令メモリ。

【請求項１２】　前記命令トランスレータは、前記命令記憶手段から読み出すべき命令のアドレスに基づいて、前記第２の命令体系の命令を前記第１の命令体系の命令に変換する処理と、前記第２の命令体系の命令をそのまま出力する処理とのいずれかを選択的に実行するための手段を含む、請求項１１に記載のトランスレータ付命令メモリ。

【請求項１３】　前記命令記憶手段からの読出時にアドレス変換を行なうためのアドレス変換手段をさらに含む、請求項１１に記載のトランスレータ付命令メモリ。

【請求項１４】　前記命令トランスレータは、

前記プロセッサが実行すべき命令の、前記トランスレータ付命令メモリにおけるアドレスを受けて前記トランスレータ付命令メモリから対応する命令を読出し、読出された前記第２の命令体系の命令を前記第１の命令体系の命令に変換する変換手段と、

前記変換手段により変換された後の前記第１の命令体系の命令を、前記トラン

スレータ付命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、

前記プロセッサが実行すべき命令のアドレスを受けて前記保持手段を探索し、前記保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、前記変換手段の出力する命令と、前記保持手段に保持されていた、対応の前記第１の命令体系の命令とを選択的に前記プロセッサに出力するための選択手段とを含む、請求項１１に記載のトランスレータ付命令メモリ。

【請求項１５】　第１の命令体系の命令をネイティブ命令として動作するプロセッサと、

前記プロセッサが接続される信号転送手段と、

前記信号転送手段を介して前記プロセッサと相互接続されるトランスレータ付命令メモリとを含み、

前記トランスレータ付命令メモリは、

前記プロセッサから前記信号転送手段を介して転送される前記第２の命令体系の命令を記憶する命令記憶手段と、

前記命令記憶手段から出力される前記第２の命令体系の命令を、前記第１の命令体系の命令に変換して前記信号転送手段を介して前記プロセッサに与えるための命令トランスレータとを含む、データ処理装置。

【請求項１６】　さらに、前記信号転送手段によって前記プロセッサと相互接続される第２の命令メモリを含み、

前記第２の命令メモリは、

前記プロセッサから前記信号転送手段を介して転送される前記第１の命令体系の命令を記憶する命令記憶手段と、

前記プロセッサから前記信号転送手段を介して与えられるアドレス信号に応答して、前記命令記憶手段から出力される前記第１の命令体系の命令を、前記信号転送手段を介して前記プロセッサに与えるための命令読出手段とを含む、請求項１５に記載のデータ処理装置。

【請求項１７】　さらに、前記トランスレータ付命令メモリから読み出される命令を前記プロセッサに転送するときの前記信号転送手段のウェイト数が、前

記第２の命令メモリから読み出される命令を前記プロセッサに転送するときの前記信号転送手段のウェイト数よりも大きくなるように前記信号転送手段による転送を制御するための転送制御手段を含む、請求項１６に記載のデータ処理装置。

【請求項１８】　さらに、前記信号転送手段によって前記プロセッサと相互接続される第３のトランスレータ付命令メモリを含み、

前記第３のトランスレータ付命令メモリは、

前記プロセッサから前記信号転送手段を介して転送される、前記第２の命令体系とは異なる第３の命令体系の命令を記憶する命令記憶手段と、

前記プロセッサから前記信号転送手段を介して与えられるアドレス信号に応答して、前記命令記憶手段から出力される前記第３の命令体系の命令を、前記第１の命令体系の命令に変換して前記信号転送手段を介して前記プロセッサに与えるための命令読出手段とを含む、請求項１６に記載のデータ処理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明はあるプロセッサにとっての非ネイティブ命令を、そのプロセッサのネイティブ命令に変換するための命令トランスレータ、そのトランスレータを備えた命令トランスレータ機能付メモリおよびそれらを用いて、非ネイティブ命令を高速に実行することが可能なデータ処理装置に関する。

【０００２】

【従来の技術】

プロセッサアーキテクチャとそのプロセッサで実行可能な命令体系とは密接な関係を持つ。一方、プロセッサアーキテクチャが進化して命令体系が新しくなると、旧命令体系で実現されたプログラムコードはそのままでは実行不可能となることが通常である。どのようにして旧命令体系で実現されたプログラム資産を有効に承継するかが問題となる。そのため、ある命令体系を持つ新プロセッサで、旧命令体系にしたがって設計された旧プロセッサ用に記述されたプログラムを実行するようにするための手法が多く開発されている。

【０００３】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するために従来行なわれている代表的な方法に、新プロセッサのハードウェアに旧プロセッサの機能を持たせる方法がある。図１を参照して、そうした方法を実現する従来のデータ処理装置５００は、旧プロセッサの命令と新プロセッサの命令との双方をデコードする機能を有する多機能命令デコーダ５およびそれら命令を実行する機能を有する演算部６を備えたプロセッサ１と、プロセッサ１に接続されるバス４と、バス４に接続されるデータメモリ２および命令メモリ３とを含む。

【０００４】

命令メモリ３は、旧プロセッサの命令と新プロセッサの命令との双方を保持する。多機能命令デコーダ５は、命令メモリ３から読出され、バス４を介してプロセッサ１に転送された命令をデコードする。このとき、多機能命令デコーダ５は、この命令が新プロセッサ用のものであっても、旧プロセッサ用のものであってもデコードすることができる。デコードされた命令は演算部６によって実行される。データメモリ２は、新プロセッサ用の命令からも、旧プロセッサ用の命令からもアクセスできる。

【０００５】

このように新プロセッサのハードウェアに旧プロセッサのハードウェアの機能を持たせる例は、たとえば「IA-32 Application Execution Model in an IA-64 System Environment」（IA-64 Application Developer & Archtecture Guide, Chapter 6, May 1999）に詳しく記載されている。

【０００６】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するための従来の他の方法として、旧プロセッサ用のソフトウェアを新プロセッサ用のソフトウェアに変換した後に実行する方法、および旧プロセッサ用の命令の動作を新プロセッサ用のソフトウェアでエミュレートする方法がある。この方法は例えば、Tom Thompsonによる「An Alpha in PC Clothing (Digital Equipment's new x86 emulator technology makes an Alpha system a fast x86 clone)」（BYTE, pp. 195-196, February 1996）に詳しく記載されている。

【０００７】

ところで、ある命令体系で記述されたプログラムを他の命令体系にしたがって設計されたプロセッサで実行することは、これ以外の場合にも有効である。たとえば、ある命令体系のサブセットを定義してその縮小された命令体系でプログラムを記述すると、プログラムサイズを小さくすることができる。また、ＪＡＶＡ言語は、仮想的なプロセッサの命令体系を定めてプログラムを記述し、その同一のプログラムを複数のプロセッサでそれぞれのプロセッサの命令体系を用いて実行する仕組みとなっている。このためＪＡＶＡ言語で記述されたプログラムは、異なる命令体系の複数種類のプロセッサで共有して実行することができる。

【０００８】

プログラムサイズを小さくする目的でサブセットの縮小命令体系を作り、プロセッサの多機能命令デコーダで非縮小命令体系の命令と縮小命令体系の命令との双方の命令をデコードする方法は既に多数提案されている。たとえばJames L. Turleyによる「Thumb Squeezes ARM Code Size (New Core Module Provides Optimized Second Instruction Set)」 (Micro Processor Report, Vol. 9, No. 4, pp. 1, 6-9, March 27, 1995) の記載を参照。

【０００９】

【発明が解決しようとする課題】

しかし、上述したような従来の手法には、いずれも以下に述べるような問題点がある。

【００１０】

複数個の命令体系によって記述されたプログラムを実行する機能をプロセッサのハードウェアに持たせる場合、ハードウェアが複雑になり、かつそのサイズが大きくなる。また、実行すべき命令体系を追加したり変更したりする場合には、ハードウェア全体を設計し直す必要があり、柔軟に対応することが困難である。

【００１１】

ソフトウェアによりプログラムを変換する場合には次のような問題がある。プログラム自体を変換する場合には、変換後プログラムを保持するために大容量のメモリを新たに必要とする。その結果、メモリのコストが増大しデータ処理装置のコストも上昇する。また命令の動作を他の命令体系の命令でエミュレートする

場合には、演算結果をエミュレートすることが必要であることはもちろん、プログラムカウンタの値、および必要がある場合にはフラグまでエミュレートしなければならない。その結果、一つの命令の動作を、別の体系の多数の命令で置換することが必要になる。その結果、動作速度が大幅に低下するという問題がある。

【００１２】

この発明はこうした問題を解決するためになされたもので、プロセッサのハードウェア自体は変更せずに、複数個の異なる命令体系の命令からなるプログラムをネイティブ命令を用いて高速に実行可能なデータ処理装置であって、大容量のメモリを必要としないデータ処理装置、そのための命令トランスレータおよび命令トランスレータ機能付メモリを提供することを目的とする。

【００１３】

【課題を解決するための手段】

請求項１に記載の発明にかかる命令トランスレータは、第１の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、第１の命令体系と異なる第２の命令体系の命令を記憶する命令メモリとともに用いられ、第１の命令体系の命令に変換してプロセッサに与えるための命令トランスレータであって、プロセッサが実行すべき命令の、命令メモリにおけるアドレスを受けて命令メモリから対応する命令を読出し、読出された第２の命令体系の命令を第１の命令体系の命令に変換するための変換手段と、変換手段により変換された後の第１の命令体系の命令を、命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、プロセッサが実行すべき命令のアドレスを受けて保持手段を探索し、保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、変換手段の出力する命令と、保持手段に保持されていた、対応の第１の命令体系の命令とを選択的にプロセッサに出力するための選択手段とを含む。

【００１４】

請求項１に記載の発明によれば、命令トランスレータを追加することにより、プロセッサ本体の構成を変更することなく非ネイティブ命令をネイティブ命令に変換してプロセッサで実行することができる。また、第１の命令体系に変換され

た命令を一時的に保持手段に保持し、次に当該命令の読出が命令メモリから行なわれるときは、保持手段に保持されていた変換済みの第１の命令体系の命令を出力することができるため、命令メモリからの読出処理と変換処理とを省略することができ、高速に変換後の命令を出力できる。

【００１５】

請求項２に記載の発明にかかる命令トランスレータは、請求項１に記載の発明の構成に加えて、第２の命令体系は可変長命令体系であり、変換手段は、命令メモリから読出された第２の命令体系の命令を、読出された第２の命令体系の命令の命令長に依存した数の第１の命令体系の命令に変換するための可変長命令変換手段を含む。

【００１６】

請求項２に記載の発明によれば、請求項１に記載の発明の作用に加えて、非ネイティブ命令の長さを変換後のネイティブ命令の個数でエミュレートするため、非ネイティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がない。

【００１７】

請求項３に記載の発明にかかる命令トランスレータは、請求項２に記載の発明の構成に加えて、可変長命令変換手段は、命令メモリから読出された第２の命令体系の命令を、読出された第２の命令体系の命令の命令長に依存し、かつそれよりも長い長さの第１の命令体系の命令に変換するための第１の手段を含む。

【００１８】

請求項３に記載の発明によれば、請求項２に記載の発明の作用に加えて、非ネイティブ命令の長さを、その非ネイティブ命令の長さに依存した変換後のネイティブ命令の長さによりエミュレートするので、非ネイティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がない。かつ変換前の非ネイティブ命令の平均長が短くなるため、プログラムサイズが小さくなりプログラムを保存するためのメモリが小さくてすむ。

【００１９】

請求項４に記載の発明にかかる命令トランスレータは、請求項３に記載の発明

の構成に加えて、第１の命令体系の各命令は１または複数個のサブ命令を含み、第１の手段が変換する第１の命令体系の命令に含まれるサブ命令の個数は、読み出された第２の命令体系の命令の長さに依存する。

【００２０】

請求項４に記載の発明によれば、請求項３に記載の発明の作用に加えて、変換後の第１の命令体系の命令に複数個のサブ命令を含ませることにより、複数個の処理を指定することができる。そのため非ネイティブ命令である第２の命令体系の命令のエミュレートを容易に行なうことができる。さらに、非ネイティブ命令の長さをサブ命令の個数に依存する第１の命令体系のネイティブ命令の長さでエミュレートできる。

【００２１】

請求項５に記載の発明にかかる命令トランスレータは、請求項１に記載の発明の構成に加えて、変換手段は、命令メモリから読出された第２の命令体系の複数個の命令を、１つの第１の命令体系の命令に変換するための複数命令変換手段を含む。

【００２２】

請求項５に記載の発明によれば、請求項１に記載の発明の作用に加えて、複数の非ネイティブ命令を１つのネイティブ命令に一度に変換することができるので、命令の変換効率がよい。

【００２３】

請求項６に記載の発明にかかる命令トランスレータは、請求項１に記載の発明の構成に加えて、第１の命令体系の各命令は、１または複数個のサブ命令を含むことが可能であり、変換手段は、命令メモリから読出された第２の命令体系の複数個の命令を、複数個の命令の数に依存した数のサブ命令を含む第１の命令体系の命令に変換するための手段を含む。

【００２４】

請求項６に記載の発明によれば、請求項１に記載の発明の作用に加えて、複数個のサブ命令によって複数個の非ネイティブ命令をエミュレートするので命令の変換が容易であり、かつ、ネイティブ命令の長さはそこに含まれるサブ命令の個

数に依存するので、非ネイティブ命令のプログラムカウンタ値を変換後のネイティブ命令の長さでエミュレートすることができる。

【００２５】

請求項７に記載の発明にかかる命令トランスレータは、請求項６に記載の発明の構成に加えて、変換後の第１の命令体系の命令に含まれるサブ命令の個数が、複数個の命令の数と等しい。

【００２６】

請求項７に記載の発明によれば、請求項６に記載の発明の作用に加えて、変換前の非ネイティブ命令の個数と変換後のサブ命令の個数とが等しいので、非ネイティブ命令とネイティブ命令のサブ命令とを対照させることにより、変換が容易に行える。

【００２７】

請求項８に記載の発明にかかる命令トランスレータは、第１の命令体系の命令をネイティブ命令として動作するプロセッサにおいて、第１の命令体系と異なる第２の命令体系の命令を記憶する命令メモリとともに用いられ、第１の命令体系の命令に変換してプロセッサに与えるための命令トランスレータであって、プロセッサが実行すべき命令の、命令メモリにおけるアドレスを受けて命令メモリから対応する命令を読出し、読出された第２の命令体系の命令を第１の命令体系の１または複数個の命令に変換する変換手段と、変換手段により変換された後の第１の命令体系の命令を、命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、プロセッサが実行すべき命令のアドレスを受けて保持手段を探索し、保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、変換手段の出力する命令と、保持手段に保持されていた、対応の、第１の命令体系の命令とを選択的にプロセッサに出力するための選択手段と、保持手段に保持されている命令を第１の条件および第２の条件のいずれかで無効化可能なエントリとして保持するよう保持手段を制御するための保持制御手段とを含む。

【００２８】

請求項８に記載の発明によれば、１つの非ネイティブ命令を１または複数個の

ネイティブ命令に変換して保持手段に保持することにより、次に同じ非ネイティブ命令の読出が行なわれるときには保持手段から変換後のネイティブ命令を高速に出力することができる。さらに、保持手段に保持されている命令の無効化に対して第１の条件および第２の条件のいずれかとすることができるので、複数のネイティブ命令の同時無効化など、無効化のための条件が複雑な場合でも容易に対処できる。

【００２９】

請求項９に記載の発明にかかる命令トランスレータは、請求項８に記載の発明の構成に加えて、第１の条件は保持手段による所定のアルゴリズムに基づくハードウェア制御による保持制御の条件であり、第２の条件は保持手段の外部からの明示的な無効化指示があったという条件である。

【００３０】

請求項９に記載の発明によれば、請求項８に記載の発明の作用に加えて、ハードウェア制御により保持手段の保持内容を維持することに加えて、外部から明示的に保持手段の内容を無効化することが可能であり、ソフトウェアの責任において安全に保持手段の保持内容を維持することができる。

【００３１】

請求項１０に記載の発明にかかる命令トランスレータは、請求項８に記載の発明の構成に加えて、保持制御手段は第２の条件で無効化が可能なエントリを無効化することなしに新たな命令を保持手段に保持できないときにアサートされる信号を出力する。

【００３２】

請求項１０に記載の発明によれば、請求項８に記載の発明の作用に加えて、新たな命令を保持手段に保持できないときには、外部に信号がアサートされる。この信号に応答して、無効化しても安全なエントリを無効化することをソフトウェア処理によって明示的に指示して、新たな命令を保持可能とすることができる。

【００３３】

請求項１１に記載の発明にかかるトランスレータ付命令メモリは、第１の命令体系の命令をネイティブ命令として動作するプロセッサとともに用いられるトラ

ンスレータ付命令メモリであって、第２の命令体系の命令を記憶する命令記憶手段と、命令記憶手段から出力される第２の命令体系の命令を、第１の命令体系の命令に変換してプロセッサに与えるための命令トランスレータとを含む。

【００３４】

請求項１１に記載の発明によれば、命令記憶手段に第２の命令体系の命令を記憶しておき、これを命令トランスレータで第１の命令体系の命令に変換してプロセッサに与えることができる。プロセッサの変更なしに、非ネイティブ命令により記述されたプログラムを実行できる。

【００３５】

請求項１２に記載の発明にかかるトランスレータ付命令メモリは、請求項１１に記載の発明の構成に加えて、命令トランスレータは、命令記憶手段から読み出すべき命令のアドレスに基づいて、第２の命令体系の命令を、第１の命令体系の命令に変換する処理と、第２の命令体系の命令をそのまま出力する処理とのいずれかを選択的に実行するための手段を含む。

【００３６】

請求項１２に記載の発明によれば、請求項１１に記載の発明の作用に加えて、第２の命令体系の命令を第１の命令体系の命令に変換して読出すことが可能であることに加え、第２の命令体系の命令をそのまま読出すことも可能なので、第２の命令体系で記述されたプログラムを他のメモリに転送したり、その内容を解析したりすることが可能になる。

【００３７】

請求項１３に記載の発明にかかるトランスレータ付命令メモリは、請求項１１に記載の発明の構成に加えて、命令記憶手段からの読出時にアドレス変換を行なうためのアドレス変換手段をさらに含む。

【００３８】

請求項１３に記載の発明によれば、請求項１１に記載の発明の作用に加えて、命令変換時と、それ以外のときとで、トランスレータ付命令メモリに関して異なるメモリマップを使用することができる。

【００３９】

請求項１４に記載の発明にかかるトランスレータ付命令メモリは、請求項１１に記載の発明の構成に加えて、命令トランスレータは、プロセッサが実行すべき命令の、トランスレータ付命令メモリにおけるアドレスを受けてトランスレータ付命令メモリから対応する命令を読出し、読出された第２の命令体系の命令を第１の命令体系の命令に変換する変換手段と、変換手段により変換された後の第１の命令体系の命令を、トランスレータ付命令メモリにおけるアドレスと関連付けて一時的に保持するための保持手段と、プロセッサが実行すべき命令のアドレスを受けて保持手段を探索し、保持手段に当該アドレスの命令に対応する命令が保持されているか否かの判定結果にしたがって、変換手段の出力する命令と、保持手段に保持されていた、対応の第１の命令体系の命令とを選択的にプロセッサに出力するための選択手段とを含む。

【００４０】

請求項１４に記載の発明によれば請求項１１に記載の発明の作用に加えて、第１の命令体系の命令に変換された命令は保持手段に保持され、次に読出が行なわれるときにその命令が保持手段に保持されていれば、あらためて命令の変換を行なうことなく、保持手段から対応の変換後の命令が出力されるので、命令記憶手段へのアクセスと、変換とに要する時間が不要となり、高速に変換後の命令を出力することができる。

【００４１】

請求項１５に記載の発明にかかるデータ処理装置は、第１の命令体系の命令をネイティブ命令として動作するプロセッサと、プロセッサが接続される信号転送手段と、信号転送手段を介してプロセッサと相互接続されるトランスレータ付命令メモリとを含み、トランスレータ付命令メモリは、プロセッサから信号転送手段を介して転送される第２の命令体系の命令を記憶する命令記憶手段と、命令記憶手段から出力される第２の命令体系の命令を、第１の命令体系の命令に変換して信号転送手段を介してプロセッサに与えるための命令トランスレータとを含む。

【００４２】

請求項１５に記載の発明によれば、プロセッサにとって非ネイティブ命令であ

る第２の命令体系の命令を、プロセッサのネイティブ命令である第１の命令体系の命令に変換する命令トランスレータが設けられているので、プロセッサの構成を変更することなく、第２の命令体系の命令で記述されたプログラムをこのプロセッサで実行することが可能となる。

【００４３】

請求項１６に記載の発明にかかるデータ処理装置は、請求項１５に記載の発明の構成に加えて、さらに、信号転送手段によってプロセッサと相互接続される第２の命令メモリを含み、第２の命令メモリは、プロセッサから信号転送手段を介して転送される第１の命令体系の命令を記憶する命令記憶手段と、プロセッサから信号転送手段を介して与えられるアドレス信号に応答して、命令記憶手段から出力される第１の命令体系の命令を、信号転送手段を介してプロセッサに与えるための命令読出手段とを含む。

【００４４】

請求項１６に記載の発明によれば、請求項１５に記載の発明の作用に加えて、第１の命令体系の命令で記述されたプログラムも第２の命令メモリからプロセッサに転送して実行できるので、プロセッサ本体の変更なしに非ネイティブ命令もネイティブ命令も区別なくプロセッサでデコードし実行することができる。

【００４５】

請求項１７に記載の発明にかかるデータ処理装置は、請求項１６に記載の発明の構成に加えて、さらに、トランスレータ付命令メモリから読み出される命令をプロセッサに転送するときの信号転送手段のウェイト数が、第２の命令メモリから読み出される命令をプロセッサに転送するときの信号転送手段のウェイト数よりも多くなるように信号転送手段による転送を制御するための転送制御手段を含む。

【００４６】

請求項１７に記載の発明によれば、請求項１６に記載の発明の作用に加えて、トランスレータ付命令メモリから読み出される命令をプロセッサに転送するときのウェイト数が多いので、その間に非ネイティブ命令からネイティブ命令への変換を行なうことができ、プロセッサでは読出対象となった命令が非ネイティブ命

令の場合もネイティブ命令の場合も区別なくフェッチしてデコードすることができる。

【００４７】

請求項１８に記載の発明にかかるデータ処理装置は、請求項１６に記載の発明の構成に加えて、さらに、信号転送手段によってプロセッサと相互接続される第３のトランスレータ付命令メモリを含み、第３のトランスレータ付命令メモリは、プロセッサから信号転送手段を介して転送される、第２の命令体系とは異なる第３の命令体系の命令を記憶する命令記憶手段と、プロセッサから信号転送手段を介して与えられるアドレス信号に応答して、命令記憶手段から出力される第３の命令体系の命令を、第１の命令体系の命令に変換して信号転送手段を介してプロセッサに与えるための命令読出手段とを含む。

【００４８】

請求項１８に記載の発明によれば、請求項１６に記載の発明の作用に加えて、トランスレータ付メモリおよび第３のトランスレータ付命令メモリは、それぞれ異なる種類の非ネイティブ命令をネイティブ命令に変換して信号転送手段に出力するので、プロセッサはどちらの非ネイティブ命令が変換された命令かを区別することなく、非ネイティブ命令で記述されたプログラムをネイティブ命令を用いて実行できる。

【００４９】

【発明の実施の形態】

図２を参照して、この発明の実施の形態のデータ処理装置は、プロセッサ１０と、プロセッサ１０に接続されたバス４０と、バス４０、ＲＥＡＤＹ信号線５０およびプロセッサ１０に接続されたバス制御回路２０と、いずれもＲＥＡＤＹ信号線５０およびバス４０に接続された、トランスレータ１４を有する圧縮命令用のトランスレータ付メモリ２４、トランスレータ１５を有するＪＡＶＡ命令用のトランスレータ付メモリ２５およびトランスレータ１６を有する非ネイティブ命令Ｘ用のトランスレータ付メモリ２６と、いずれもバス４０に接続された、ネイティブ命令用ＲＡＭ２１、データ用メモリ２２および、ネイティブ命令と、圧縮命令と、ＪＡＶＡ命令と、非ネイティブ命令Ｘと、データとを格納するＲＯＭ２

３とを含む。

【００５０】

バス制御回路２０は、プロセッサ１０からバス４０に出力されたアドレスをデコードしてネイティブ命令用ＲＡＭ２１、データ用メモリ２２ 、ＲＯＭ２３、圧縮命令用のトランスレータ付メモリ２４、ＪＡＶＡ命令用のトランスレータ付メモリ２５および非ネイティブ命令Ｘ用のトランスレータ付メモリ２６に対してこれらのいずれかをアクティベイトするチップセレクト信号ＣＳを出力する。バス制御回路２０はまた、トランスレータ１４～１６に対して、これらの命令変換機能を制御する変換機能イネーブル信号ＴＥを出力してトランスレータ１４～トランスレータ１６に与える。またバス制御回路２０は、トランスレータ１４～１６からＲＥＡＤＹ信号線５０を介して与えられるＲＥＡＤＹ信号を受け、バスサイクルの終了を示すＤＣ信号５１をプロセッサ１０に与える。

【００５１】

図３を参照して、プロセッサ１０は、コア１００と、命令キャッシュ１０１と、データキャッシュ１０２と、バス４０およびＤＣ信号５１に接続されたバスインターフェイス部１０３と、これらを互いに接続する命令アドレスバス１０４および命令バス１０５と、コア１００、命令キャッシュ１０１およびバスインターフェイス部１０３を相互に接続してアドレスおよびデータを送信するためのデータアドレスバス１０６およびデータバス１０７とを含む。

【００５２】

コア１００は、ＶＬＩＷ（Very Long Instruction Word）型命令体系を有するプロセッサである。コア１００は、命令バス１０５から入力されたＶＬＩＷ命令をデコードするための命令デコーダ１１０と、命令デコーダ１１０によってデコードされた命令を実行するためのメモリ演算部１３０および整数演算部１４０と、メモリ演算部１３０および整数演算部１４０に複数のバスで接続されたレジスタファイル１２０とを含む。

【００５３】

命令デコーダ１１０は二つのサブ命令デコーダ１１１および１１２を含む。

メモリ演算部１３０は、アドレス演算器１３１、ＰＣ演算器１３２、シフタ１

３３およびＡＬＵ１３４などの演算器を含む。メモリ演算部１３０は、サブ命令デコーダ１１１の出力にしたがい、メモリアクセス命令、ＰＣ制御命令、整数演算命令などを実行するためのものである。整数演算部１４０は、シフタ１４１、ＡＬＵ１４２、乗算器１４３およびアキュムレータ１４４を含む。整数演算部１４０は、サブ命令デコーダ１１２の出力にしたがい整数演算命令を実行するためのものである。メモリ演算部１３０および整数演算部１４０は、２つのサブ命令を並列に実行する場合と、それぞれ独立に１つのサブ命令を実行する場合とがある。

【００５４】

図４を参照して、プロセッサ１０が有するレジスタファイル１２０は、６４本の汎用レジスタであるレジスタ１５０～１５２、１６２、１６３ａ、および１６３ｂを含む。プロセッサ１０はさらに、制御レジスタ１７０～１８０を含む。また、図３に示すアキュムレータ１４４はアキュムレータ１４４ａおよびアキュムレータ１４４ｂを含む。

【００５５】

レジスタ１５０は常にゼロを保持するレジスタである。レジスタ１６２は非割込処理中のスタックトップのデータを保持するためのものである。レジスタ１６３ｂは非割込処理中のスタックポインタでスタックトップのすぐ下のデータのアドレスを保持するためのものである。レジスタ１６３ａおよび１６３ｂは、ＰＳＷ（プロセッサステータスワード）である制御レジスタ１７０中にあるモードビットで切り替わり、割込処理中はレジスタ１６３ａが使用され、非割込処理中はレジスタ１６３ｂが使用される。

【００５６】

制御レジスタ１７０～１８０は、それぞれ所定の要素のための専用のレジスタである。たとえば制御レジスタ１７０はＰＳＷであって、演算により変化するフラグ、割込処理処理中か否か、割込マスク中か否か、デバック中か否かなど、プロセッサ１０の動作モードを示すモードビットを含む。制御レジスタ１７２はプログラムカウンタ（ＰＣ）であり、現在実行中の命令のアドレスを示す。制御レジスタ１７１および１７３は、割込受付時、例外発生時、トラップ発生時にそれ

ぞれ制御レジスタ１７０および１７２の値をコピーして保持するためのものである。

【００５７】

アキュムレータ１４４ａおよび１４４ｂは、乗算結果、積和演算結果を保持するためのものである。アキュムレータ１４４ａおよび１４４ｂはそれぞれ、汎用レジスタの２倍のビット長である６４ビットのデータを保持することができる。

【００５８】

図５を参照して、制御レジスタ１７０が保持するＰＳＷは３２ビットであって、割込処理中か非割込処理中かを示すモードビットであるＳＭビット１７０ａと、割込許可中か割込禁止中かを示すＩＥビット１７０ｂと、命令の実行条件を制御するＦ０ビット１７０ｃおよびＦ１ビット１７０ｄとを含む。この他に制御レジスタ１７０は、ＲＰビット、ＭＤビット、Ｆ２～Ｆ７の各ビットを含む。これらの意味については図５に示す通りである。

【００５９】

図６を参照して、プロセッサ１０は、命令を以下のようにしてパイプライン処理する。プロセッサ１０は、メモリ演算部１３０と整数演算部１４０とで行なわれるサブ命令をそれぞれ実行するためのＭＵパイプ１３９およびＩＵパイプ１４９を含む。これらパイプはいずれも、命令フェッチステージ１９１、デコードおよびアドレス計算ステージ１９２、演算およびメモリアクセスステージ１９３およびライトバックステージ１９４からなる。

【００６０】

命令フェッチステージ１９１は、命令をフェッチして命令デコーダ１１０中の命令レジスタ１１３に保持するステージである。デコードおよびアドレス計算ステージ１９２では、この命令がサブ命令デコーダ１１１、１１２でデコードされ、同時にレジスタファイル１２０がアクセスされてオペランドおよびＰＣのアドレス計算が行なわれる。演算およびメモリアクセスステージ１９３では、整数演算およびデータメモリアクセス処理が行なわれる。ライトバックステージ１９４では、演算結果およびメモリからフェッチされたデータがレジスタファイル１２０に再び書込まれる。

【００６１】

図７を参照して、プロセッサ１０の命令２００は２ウェイのＶＬＩＷ型命令であり、図示されるようなフォーマットを有する。すなわち、命令２００は、各サブ命令の実行順序と長いサブ命令を定義するＦＭフィールド２０１ａ、２０１ｂと、サブ命令を格納するＬコンテナ２０５およびＲコンテナ２０６と、各サブ命令の実行条件を指定する条件フィールド２０３および２０４とを含む。

【００６２】

条件フィールド２０３および２０４は、制御レジスタ１７０であるＰＳＷ中のフラグＦ０およびＦ１（Ｆ０ビット１７０ｃおよびＦ１ビット１７０ｄ）の値に依存した条件を指定する。たとえば条件フィールド２０３が「０００」のとき、Ｌコンテナ２０５に含まれるサブ命令は無条件に実行される。条件フィールド２０４が「１０１」のとき、Ｒコンテナ２０６に含まれるサブ命令はＦ０＝１かつＦ１＝１のとき実行され、フラグＦ０およびＦ１（Ｆ０ビット１７０ｃおよびＦ１ビット１７０ｄ）がそれ以外の値をとる場合は無効化される。

【００６３】

ＦＭフィールド２０１ａおよび２０１ｂは、Ｌコンテナ２０５とＲコンテナ２０６とに含まれるサブ命令を実行する場合の実行動作を指定する。実行動作としては４つある。１番目は、Ｌコンテナ２０５とＲコンテナ２０６とに含まれるサブ命令を並列に実行する動作である。２番目は、Ｌコンテナ２０５のサブ命令をまず実行して次にＲコンテナ２０６のサブ命令を実行する動作である。３番目は２番目の動作の逆であり、Ｒコンテナ２０６のサブ命令をまず実行して次にＬコンテナ２０５のサブ命令を実行する動作である。４番目はＬコンテナ２０５とＲコンテナ２０６とに分割して保持された１つの長いサブ命令を実行する動作である。すなわち、ＦＭフィールド２０１ａおよびＦＭフィールド２０１ｂの値によって、上述した４つの動作のいずれかが選択される。

【００６４】

図８を参照して、Ｌコンテナ２０５およびＲコンテナ２０６に保持されるサブ命令は以下のようなフォーマットを有する。サブ命令は２８ビット長の短い命令と５４ビット長の長いサブ命令とに分類される。短い命令は、フォーマット２１

１～２１７に示される７種類のフォーマットを有する。短い命令フォーマットの概略をいえば、ビット位置０－９で演算の種類が示され、ビット位置１０－２７で最大３つのオペランドが指定される。長いサブ命令はフォーマット２１８に示されるようにビット位置０－９で演算の種類が示され、ビット位置１０－５３で３２ビット長の即値データを含む最大３つのオペランドが指定される。なお、長いサブ命令の３２ビットの即値はＶＬＩＷ命令ビット位置２６－３１、３６－４３、および４６－６３に保持される。

【００６５】

フォーマット２１１は、メモリアクセス演算（ロード／ストア演算）を行なうサブ命令のフォーマットである。フォーマット２１２は汎用レジスタに保持されたオペランド間の演算（ＯＰ演算）を行なうサブ命令のフォーマットである。フォーマット２１３－２１７は分岐演算を行なうサブ命令のフォーマットである。長いサブ命令のフォーマット２１８は、上記した３種類の演算全てに共通で使用される。

【００６６】

サブ命令を図６に示すようにプロセッサ１０でパイプライン処理する場合には、ＯＰ演算、ロード／ストア演算、分岐演算のサブ命令はそれぞれ図９に示す４段のパイプライン２２１－２２３によって示されるように４段のパイプラインステージで実行される。

【００６７】

ＦＭフィールド２０１ａおよび２０１ｂによりサブ命令の実行順序が指定されたときには、サブ命令はＭＵパイプ１３９およびＩＵパイプ１４９によって図１０に示すようにパイプライン処理される。ここで、ストールステージ２３４－２３６は、ＦＭフィールド２０１ａおよび２０１ｂの値にしたがってサブ命令に順序をつけて実行する場合に、一方のサブ命令の実行を遅延させるために挿入されるパイプラインステージである。

【００６８】

次に、プロセッサ１０に対して定義されたサブ命令の一覧を示す。この一覧において、各サブ命令のニーモニックを大文字で、その処理内容を各ニーモニック

の右側に、それぞれ示す。

【００６９】

ロード／ストア命令

LDB	符号拡張してレジスタに１バイトをロード
LDBU	ゼロ拡張してレジスタに１バイトをロード
LDH	符号拡張してレジスタに半ワードをロード
LDHH	レジスタ上位に半ワードをロード
LDHU	ゼロ拡張してレジスタに半ワードをロード
LDW	レジスタに１ワードをロード
LD2W	２つのレジスタに２ワードをロード
LD4BH	符号拡張して４つの半ワードレジスタに４つのバイトをロード
LD4BHU	ゼロ拡張して４つの半ワードレジスタに４つのバイトをロード
LD2H	２つのレジスタに２つの半ワードをロード
STB	レジスタから１バイトをストア
STH	レジスタから半ワードをストア
STHH	レジスタ上位から半ワードをストア
STW	レジスタから１ワードをストア
ST2W	レジスタから２ワードをストア
ST4HB	４つの半ワードレジスタから４つのバイトをストア
ST2H	２つのレジスタから２半ワードをストア
MODDEC	５ビット即値によってレジスタ値をデクリメント
MODINC	５ビット即値によってレジスタ値をインクリメント

転送命令

MVFSYS	制御レジスタから汎用レジスタに転送
MVTSYS	汎用レジスタから制御レジスタに転送
MVFACC	アキュムレータから１ワードを転送

MVTACC　　２つの汎用レジスタから２ワードをアキュムレータに転送

比較命令

CMPcc　　比較

cc= EQ(000),NE(001),GT(010),GE(011),LT(100),
LE(101),PS-both positive(110), NG-both negative(111)

CMPUcc　　符号なし比較

cc=GT(010),GE(011),LT(100),LE(101)

算術演算命令

ABS　　絶対値

ADD　　加算

ADDC　　キャリー付加算

ADDHppp　　半ワード加算

ppp=LLL(000),LLH(001),LHL(010),LHH(011),
HLL(100),HLH(101),HHL(110),HHH(111)

ADDS　　第３オペランドの符号をレジスタRbに加算

ADDS2H　　２つの半ワードに符号を加算

ADD2H　　２対の半ワードを加算

AVG　　正の無限大方向の丸めによる平均

AVG2H　　正の無限大方向の丸めにより２対の半ワードを平均

JOINpp　　２つの半ワードをジョイン

pp=LL(00),LH(01),HL(10),HH(11)

SUB　　減算

SUBB　　ボロー付の減算

SUBHppp　　半ワードの減算

ppp=LLL(000),LLH(001),LHL(010),LHH(011),

HLL(100),HLH(101),HHL(110),HHH(111)

SUB2H	２対の半ワードの減算	

論理演算命令

AND	論理AND	
OR	論理OR	
NOT	論理NOT	
XOR	排他的論理OR	
ANDFG	フラグの論理AND	
ORFG	フラグの論理OR	
NOTFG	フラグの論理NOT	
XORFG	フラグの排他的論理OR	

シフト演算命令

SRA	右に算術シフト	
SRAHp	右に半ワードを算術シフト	p=L(0),H(1)
SRA2H	２つの半ワードを右に算術シフト	
SRC	レジスタを連接して右にシフト	
SRL	右に論理シフト	
SRLHp	半ワードを右に論理シフト	p=L(0),H(1)
SRL2H	２つの半ワードを右に論理シフト	
ROT	右ローテイト	
ROT2H	２つの半ワードを右ローテイト	

ビット演算命令

BCLR	ビットをクリア	
BNOT	ビットを反転	
BSET	ビットをセット	
BTST	ビットをテスト	

分岐命令

BRA	分岐
BRATZR	ゼロなら分岐
BRATNZ	ゼロでなければ分岐
BSR	サブルーチンに分岐
BSRTZR	ゼロならサブルーチンに分岐
BSRTNZ	ゼロでなければサブルーチンに分岐
DBRA	遅延分岐
DBRAI	即値付遅延分岐
DBSR	サブルーチンへの遅延分岐
DBSRI	サブルーチンへの即値付遅延分岐
DJMP	遅延ジャンプ
DJMPI	即値付遅延分岐
DJSR	サブルーチンへの遅延ジャンプ
DJSRI	サブルーチンへの即値付遅延ジャンプ
JMP	ジャンプ
JMPTZR	ゼロならジャンプ
JMPTNZ	ゼロでなければジャンプ
JSR	サブルーチンに分岐
JSRTZR	ゼロならサブルーチンにジャンプ
JSRTNZ	ゼロでなければサブルーチンにジャンプ
NOP	ノーオペレーション

OS関連命令

TRAP	トラップ
REIT	例外、割込およびトラップからの復帰

DSP算術演算命令

MUL	乗算	
MULX	拡張精度の乗算	
MULXS	拡張精度での乗算および左へのシフト	
MULX2H	拡張精度で２対の半ワードを乗算	
MULHXpp	拡張精度で２つの半ワードを乗算	
	pp=LL(00),LH(01),HL(10),HH(11)	
MUL2H	２対の半ワードを乗算	
MACd	乗算および加算	(d=0,1)
MACSd	乗算、左１ビットシフト、加算	(d=0,1)
MSUBd	乗算および減算	(d=0,1)
MSUBSd	乗算、左１ビットシフト、減算	(d=0,1)
SAT	サチュレート	
SATHH	ワードオペランドを上位半ワードにサチュレート	
SAHL	ワードオペランドを下位半ワードにサチュレート	
SATZ	正数にサチュレート	
SATZ2H	２つの半ワードを正数にサチュレート	
SAT2H	２つの半ワードオペランドをサチュレート	

繰返し命令

REPEAT	１ブロックの命令を繰返す
REPEATI	１ブロックの命令を即値付繰返す

デバッガサポート命令

DBT	デバッグトラップ
RTD	デバッグ割込およびトラップからの復帰

図１１に、図２に示すプロセッサ１０がバス４０を介してネイティブ命令用ＲＡＭ２１、データ用メモリ２２、ＲＯＭ２３、圧縮命令用のトランスレータ付メモリ２４、ＪＡＶＡ命令用のトランスレータ付メモリ２５および非ネイティブ命

令X用のトランスレータ付メモリ２６をアクセスする場合のアドレスマップを示す。図１１に示されるように、ネイティブ命令用ＲＡＭ２１、データ用メモリ２２およびＲＯＭ２３はそれぞれ、ネイティブ命令用アドレス領域１２１、データメモリ用アドレス領域１２２およびＲＯＭ用アドレス領域１２３にマップされる。圧縮命令用のトランスレータ付メモリ２４、ＪＡＶＡ命令用のトランスレータ付メモリ２５および非ネイティブ命令X用のトランスレータ付メモリ２６はそれぞれ、命令コードの変換を行なわずにリードライトする場合には圧縮命令（直結）用アドレス領域１２４ａ、ＪＡＶＡ命令用（直結）アドレス領域１２５ａおよび非ネイティブ命令Ｘ（直結）用アドレス領域１２６ａにマップされる。圧縮命令用のトランスレータ付メモリ２４、ＪＡＶＡ命令用のトランスレータ付メモリ２５および非ネイティブ命令X用のトランスレータ付メモリ２６は、命令コードの変換を伴ってリードする場合にはそれぞれ圧縮命令用領域（トランスレータ経由）１２４ｂ、ＪＡＶＡ命令（トランスレータ経由）用アドレス領域１２５ｂ、およびアドレス領域１２６ｂにそれぞれマップされる。

【００７０】

図１２を参照して、圧縮命令用のトランスレータ付メモリ２４は、トランスレータ１４と圧縮命令用ＲＡＭ２４５とを含む。トランスレータ１４はトランスレーション回路２４３と、入力された値を１ビット右シフトして１／２の値としてアドレス線２５３に出力するためのアドレス変換器２４１と、ＴＥ信号線２５０上の変換機能イネーブル信号ＴＥが命令の伸張を指示している場合に、アドレス線２５３を選択して値が１／２にされたアドレスを選択し、それ以外の場合にはアドレス線２５２の値を選択して、圧縮命令用ＲＡＭ２４５とトランスレーション回路２４３とに与えるためのＭＵＸ２４２と、圧縮命令用ＲＡＭ２４５へデータ線２５９から命令を書込むか、読出すかを指定するＲ／W信号２５１とアドレス線２５２上のアドレスとに基づいて、圧縮命令用ＲＡＭ２４５に対して書込みがあると、トランスレーション回路２４３中にあるキャッシュメモリの、アドレス線２５２上のアドレスに対応するエントリを無効化することを指示する制御信号２５７ａを出力してトランスレーション回路２４３に与えるための書込検出回路２４６とを含む。制御信号２５７ａがアサートされると、トランスレーション

回路２４３はそのときのアドレス線２５４上のアドレスを見て、対応するキャッシュのエントリを無効化する。これにより、圧縮命令用ＲＡＭ２４５の古い内容に基づく古い変換結果がキャッシュから読出されることを防ぐ。また、アドレス変換器２４１により、圧縮命令用のトランスレータ付メモリ２４へのアクセスアドレスが変更される。

【００７１】

圧縮命令用のトランスレータ付メモリ２４ はさらに、圧縮命令用のトランスレータ付メモリ２４から命令コードを読出す場合に、圧縮命令用ＲＡＭ２４５から命令コード出力線２５７に出力された命令コードをそのままデータ線２５９へ出力するか、それをトランスレーション回路２４３で伸張した後の命令コードを伸張後命令コード出力線２５８からデータ線２５９へ出力するかのいずれか一方を選択するためのＭＵＸ２４４を含む。トランスレーション回路２４３はまた、伸張後の命令コードを出力するタイミングをバス制御回路２０へのＲＥＡＤＹ信号線５０に与えるための、トライステート信号であるＲＥＡＤＹ信号をＲＥＡＤＹ信号線２５６上に出力する。

【００７２】

プロセッサ１０から見ると、 圧縮命令用のトランスレータ付メモリ２４のアドレスは圧縮命令（直結）用アドレス領域１２４ａと圧縮命令用領域（トランスレータ経由）１２４ｂとからなる。圧縮命令（直結）用アドレス領域１２４ａで圧縮命令用のトランスレータ付メモリ２４をアクセスする場合、６４ｋBの空間の全バイト位置に有効なメモリがある。一方、圧縮命令用領域（トランスレータ経由）１２４ｂで圧縮命令用のトランスレータ付メモリ２４をアクセスする場合、１２８ｋBの空間のアラインされた各８バイトの上位４バイトには有効なメモリがあるが、下位４バイトには有効なメモリは存在しない。圧縮命令用のトランスレータ付メモリ２４では、もともと８バイトのＶＬＩＷ命令の中の２つのサブ命令を、アドレスを変えずにそれぞれ２バイトに圧縮して保存しているためである。プロセッサ１０がたとえばアドレスH’ ２００２０１００から圧縮されたＶＬＩＷ命令をフェッチして実行するとき、圧縮命令用のトランスレータ付メモリ２４からは２バイトの命令２つが１つの８バイトのＶＬＩＷ命令に伸張されて出

力され、プロセッサ１０ではＰＣ値を８バイト進めてアドレスH’２００２０１０８から次の命令をフェッチして実行する。

【００７３】

図１３を参照して、トランスレーション回路２４３は、アドレス線２５４から与えられるアドレスに対応する伸張後の命令を保持している場合には伸張後のＶＬＩW命令を出力線３５９に出力し、あわせてキャッシュヒット信号線３５７上のキャッシュヒット信号をアサートするキャッシュメモリ３５４と、命令コード出力線２５７から入力された２つの命令を８バイトのＶＬＩW命令に伸張して出力線３６０に出力し、あわせて命令の伸張が終了したタイミングを示す信号をタイミング信号線３５８に出力するための命令コード伸張部３５０と、キャッシュメモリ３５４からの出力線３５９と命令コード伸張部３５０からの出力線３６０との一方を選択して出力するためのＭＵＸ３５６と、キャッシュメモリ３５４からのキャッシュヒット信号線３５７と命令コード伸張部３５０からのタイミング信号線３５８との論理和を出力するためのＯＲ回路３５５とを含む。命令コード伸張部３５０は、アドレス線２５４上のアドレスに基づいて、命令コード出力線２５７から入力された２つの圧縮された命令の一方を選択して出力するためのＭＵＸ３５１と、ＭＵＸ３５１の出力する命令を伸張する伸張器３５２とを含む。

【００７４】

アドレス線２５４から入力されたアドレスはキャッシュメモリ３５４ と命令コード伸張部３５０とへ入力される。キャッシュメモリ３５４は後述するように１エントリが８バイトで容量２ｋＢの２ウェイセットアソシアティブキャッシュである。キャッシュメモリ３５４は、入力されたアドレスと同一アドレスの命令が過去に伸張されて保持されているか否かを調べ、保持されている場合には伸張後のＶＬＩW命令を出力線３５９に出力するとともに、キャッシュメモリ３５４がヒットしたことを示すためにキャッシュヒット信号線３５７上のキャッシュヒット信号をアサートする。このキャッシュヒット信号はＯＲ回路３５５を経てＲＥＡＤＹ信号線２５６上のＲＥＡＤＹ信号をアサートする。このとき、ＭＵＸ３５６は出力線３５９上の出力を選択してキャッシュメモリ３５４から出力された伸張後のＶＬＩW命令をデータ線２５９上に出力する。

【００７５】

一方、キャッシュメモリ３５４がミスしたときには、命令コード伸張部３５０から出力線３６０に出力された伸張後の８バイトのＶＬＩＷ命令がＭＵＸ３５６によって選択され、データ線２５９に出力される。このとき、データ線２５９に出力された伸張後の命令コードは、キャッシュメモリ３５４にも転送され、対応するエントリに書込まれる。伸張器３５２での命令の伸張が終了したタイミングで、タイミング信号線３５８上のタイミング信号がアサートされる。このタイミング信号は、ＯＲ回路３５５を経てＲＥＡＤＹ信号線２５６上のＲＥＡＤＹ信号をアサートする。

【００７６】

図１４に、キャッシュメモリ３５４の詳細を示す。図１４に示されるようにキャッシュメモリ３５４は２ウェイセットアソシアティブキャッシュであって、アドレス線２５４のうちの下位７ビットをインデックスとして、各ウェイに保持されているタグおよび命令コードをインデックスに基づいてサーチし、一致するタグがあるか否かを判定する。一致するタグがある場合には一致信号がそのウェイから出力される。この一致信号がＯＲ回路を経てキャッシュヒット信号線３５７上にキャッシュヒット信号として出力される。一方、このとき、２つのウェイの、アドレス線２５４のインデックスにより示されるエントリからは対応の命令コード（６４ビット）が出力され、一致信号に基づいてそのうちの一つが選択され出力バッファを経て出力線３５９上に出力される。

【００７７】

図１５に、プロセッサ１０が圧縮命令用のトランスレータ付メモリ２４ をアクセスして命令をフェッチするときのバス４０、ＲＥＡＤＹ信号線５０、ＤＣ信号５１の信号変化を説明するタイミングチャートを示す。図１５を参照して、キャッシュメモリ３５４がヒットしたとき、ＲＥＡＤＹ信号線５０上のＲＥＡＤＹ信号がバスサイクル開始後１クロック後にアサートされ、バスサイクル開始の２クロック後にバス制御回路２０からＤＣ信号５１によりバスサイクル終了がプロセッサ１０に通知される。また、キャッシュメモリ３５４に保持されていた伸張後のＶＬＩＷ命令の命令コードがバスサイクル開始後１．５クロック後に圧縮命

令用のトランスレータ付メモリ２４からデータ線２５９を経由してバス４０に出力され、ＤＣ信号５１のサンプリングタイミングと同じバスサイクル開始後２クロック後にプロセッサ１０に取込まれる。

【００７８】

一方、キャッシュメモリ３５４がミスしたときには、ＲＥＡＤＹ信号線５０がバスサイクル開始後１クロック後に一度ネゲートされ、バスサイクル開始後の３クロック後にアサートされる。これによりバスサイクル開始の２および３クロック後はバス制御回路２０からバスサイクルにウェイトサイクル５２および５３を挿入することがＤＣ信号５１によりプロセッサ１０に通知される。バスサイクル開始の４クロック後にバス制御回路２０からＤＣ信号５１によりバスサイクル終了がプロセッサ１０に通知される。また、　命令コード伸張部３５０ により伸張されたＶＬＩＷ命令の命令コードがバスサイクル開始３．５クロック後に圧縮命令用のトランスレータ付メモリ２４からデータ線２５９ 経由でバス４０に出力され、ＤＣ信号５１のサンプリングタイミングと同じくバスサイクル開始後４クロック後にプロセッサ１０に取込まれる。

【００７９】

図１６に、１つの４バイト境界にある圧縮された二つの命令３０５、３０６が伸張後のＶＬＩＷ命令のサブ命令のフィールドと他のフィールドとにどのように対応するかを示す。図１６を参照して、圧縮された命令３０５および命令３０６との実行順序に関して、そのアドレス位置により論理的にまず命令３０５が実行され、次に命令３０６が実行される。伸張器３５２は、命令３０５と命令３０６との内容に依存して、プロセッサ１０の命令３０５に対するサブ命令をＬコンテナ２０５またはＲコンテナ２０６のフィールドに発生し、残ったコンテナに命令３０６に対応するサブ命令を発生する。つまり、２つの命令から１つの命令が発生される。ＦＭフィールド２０１ａおよび２０１ｂは命令３０５と命令３０６とのオペランドに依存関係がなくかつプロセッサ１０で２つの命令が並列実行可能なときは「００」となる。オペランドに依存関係があるか、またはプロセッサ１０のハードウェア構成上並列実行不可能な場合、命令３０５に対応する命令がＬコンテナ２０５に入るときは「１０」、Ｒコンテナ２０６に入るとき「０１」の

値となる。

【００８０】

条件フィールド２０３および２０４はそれぞれ、命令３０５および命令３０６が条件分岐命令ＢＲＡＴか、条件分岐命令ＢＲＡＦか、それ以外の命令かにしたがって「０００」、「００１」および「０１０」の値となる。

【００８１】

図１７～図２０に、圧縮された命令がサブ命令に伸張されるときの命令ビットパターンの対応例を示す。図１７bに示す。「Add Ra, Rc」命令が伸張されるとサブ命令「ADD, Rx, Ry, Rz」となる。９ビットのオペコード３１１は異なる９ビットのオペコード３２１に変換される。４ビットのレジスタ番号Ｒａ３１２は上位に「１１」を付加してレジスタ番号Ｒｘ３２２およびレジスタ番号Ｒｙ３２３ に変換される。同様に４ビットのレジスタ番号Ｒｃ３１４はレジスタ番号Ｒｚ３２４に変換される。

【００８２】

図１８を参照して、命令「SUB Ra, #imm:4」が伸張されるとサブ命令「SUB Rx, Ry, #imm:6」になる。オペコード３１１とレジスタ番号Ｒａ３１２との変換は図１７に示す場合と同様である。４ビットのリテラル３１５は符号拡張され６ビットのリテラル３２５に変換される。

【００８３】

図１９を参照して、命令「BRA #imm:9」が伸張されるとサブ命令「BRA #imm:18」になる。オペコード３１１がオペコード３２１に変換され、９ビットの変位３１６は符号拡張されて１８ビットの変位３２６に変換される。

【００８４】

図２０を参照して、命令「LDW Ra, @(Rb+)」が伸張されるとサブ命令「LDW Rx, @(Ry+, R0)」になる。オペコード３１１がオペコード３２１に、レジスタ番号Ｒａ３１２は上位に「１１」が付加されてレジスタ番号Ｒｘに、それぞれ変換される。レジスタ番号Ｒｂ３１４は「ADD Ra, Rc」命令の場合と異なり上位に「００」が付加されてレジスタ番号Ｒｙに変換される。サブ命令のレジスタ番号３２７はＲ０を示す番号「０００００」となる。

【００８５】

以上のようにサブ命令では３オペランドが基本であるのに対し、圧縮された命令では２オペランドが基本である。サブ命令はオペランドとして６ビットフィールドで６４個の汎用レジスタを指定できるが、圧縮された命令ではオペランドとして４ビットのフィールドで１６個の汎用レジスタしか指定できない。また、変換時にはレジスタ番号の上位に「１１」を付加してレジスタ番号を拡張するので、圧縮された命令でオペランドとして指定できる汎用レジスタは図２１に示すように、サブ命令の汎用レジスタＲ４８～６３（１５３～１５５、…、１６２、１６３ａ、１６３ｂ）に相当する。

【００８６】

図２２を参照して、ＪＡＶＡ命令用のトランスレータ付メモリ２５は、トランスレータ１５とＪＡＶＡ命令用ＲＡＭ２６５とを含む。

【００８７】

トランスレータ１５ は、ＪＡＶＡ命令のためのトランスレーション回路２６３と、アドレス線２７２から与えられるアドレスを右に３ビットシフトし、１／８の値としてアドレス線２７３に出力するためのシフト回路２６１と、変換機能制御信号線２７０上の変換機能イネーブル信号ＴＥの値に基づいて、アドレス線２７３およびアドレス線２７２の一方を選択し、アドレス線２７４を介してＪＡＶＡ命令のためのトランスレーション回路２６３およびＪＡＶＡ命令用ＲＡＭ２６５に与えるためのＭＵＸ２６２と、出力線２７７および出力線２７８の一方を選択してデータ線２７９に出力するためのＭＵＸ２６４と、Ｒ／Ｗ信号線２７１およびアドレス線２７２に接続され、ＪＡＶＡ命令用ＲＡＭ２６５に対する書込があったことを検出してＪＡＶＡ命令のためのトランスレーション回路２６３中にあるキャッシュメモリの全エントリを無効化することを指示する制御信号を制御信号線２７７ａ 上に出力するための書込検出回路２６６とを含む。

【００８８】

ＭＵＸ２６２ は、変換機能イネーブル信号ＴＥ が命令変換を指示している場合に、アドレス線２７３ を選択して値が１／８にされたアドレスをアドレス線２７４を介してＪＡＶＡ命令用ＲＡＭ２６５およびＪＡＶＡ命令のためのトラン

スレーション回路２６３に与える。またＭＵＸ２６２は、ＪＡＶＡ命令のためのトランスレーション回路２６３　が命令コードの変換を行なうためにＪＡＶＡ命令用ＲＡＭ２６５を２回目以降アクセスするとき、ＪＡＶＡ命令のためのトランスレーション回路２６３からアドレス線２７５を介して与えられたアドレスをアドレス線２７４に出力する。

【００８９】

ＭＵＸ２６４は、ＪＡＶＡ命令用のトランスレータ付メモリ２５から命令コードを読出す場合、ＪＡＶＡ命令用ＲＡＭ２６５から出力された出力線２７７をそのままデータ線２７９ に出力するか、それをＪＡＶＡ命令のためのトランスレーション回路２６３で変換した後の、出力線２７８ を介して与えられる命令コードをデータ線２７９に出力するかを制御する。

【００９０】

ＲＥＡＤＹ信号線２７６上のＲＥＡＤＹ信号は、ＪＡＶＡ命令のためのトランスレーション回路２６３から変換の後の命令コードを出力するタイミングを示す。ＲＥＡＤＹ信号 は、ＲＥＡＤＹ信号線５０に伝えられ、さらにバス制御回路２０に与えられるトライステート出力信号である。

【００９１】

書込検出回路２６６に与えられるＲ／Ｗ信号線２７１上のＲ／Ｗ信号は、ＪＡＶＡ命令用ＲＡＭ２６５へデータ線２７９から命令コードを書込むか、読出すかを指定する信号である。ＪＡＶＡ命令用ＲＡＭ２６５に対する書込があると、書込検出回路２６６はそれを検知して制御信号線２７７ａ上に制御信号を出力する。この制御信号がアサートされると、ＪＡＶＡ命令のためのトランスレーション回路２６３では全てのキャッシュエントリを無効化する。この動作により、ＪＡＶＡ命令用ＲＡＭ２６５の古い内容に基づく古い変換結果がキャッシュから読み出されることが防止できる。

【００９２】

プロセッサ１０から見ると、ＪＡＶＡ命令用のトランスレータ付メモリ２５のアドレスは図１１に示すＪＡＶＡ命令用（直結）アドレス領域１２５ａ とＪＡＶＡ命令（トランスレータ経由）用アドレス領域１２５ｂ とからなる。ＪＡＶ

Ａ命令用（直結）アドレス領域１２５ａでＪＡＶＡ命令用のトランスレータ付メモリ２５ をアクセスする場合、６４ｋＢの空間の全バイト位置に有効なメモリがある。一方、ＪＡＶＡ命令（トランスレータ経由）用アドレス領域１２５ｂでＪＡＶＡ命令用のトランスレータ付メモリ２５をアクセスする場合、５１２ｋＢの空間のアラインされた各８バイトの上位１バイトには有効なメモリがあるが、下位７バイトには有効なメモリは存在しない。

【００９３】

プロセッサ１０ がたとえばアドレスH’ ２００５０３００から圧縮されたＶＬＩＷ命令をフェッチして実行するとき、ＪＡＶＡ命令用のトランスレータ付メモリ２５からはＪＡＶＡ命令が１または複数の８バイトのＶＬＩＷ命令に伸張されて出力され、プロセッサ１０はＪＡＶＡ命令１バイトにつきＰＣ値を８バイト進めて１または複数のＶＬＩＷ命令を実行する。なお、ＪＡＶＡ命令については Tim Lindholm and Frank Yellin, "The Java Virtual Machine Specification Second Edition," Sun Microsystems, Inc., 1999に詳しく記載されている。

【００９４】

図２３を参照して、ＪＡＶＡ命令のためのトランスレーション回路２６３は、命令コード変換部３７０と、キャッシュメモリ３７４と、アドレス線２７４から入力されたアドレスを命令コード変換部３７０およびキャッシュメモリ３７４に与え、さらにアドレス線２７５上に出力するためのアドレス制御回路３７３と、キャッシュメモリ３７４の出力線３７９および命令コード変換部３７０の出力線３８０を受け、いずれか一方から与えられる命令コードを選択してデータ線２７９に出力するためのＭＵＸ３７６と、キャッシュメモリ３７４からキャッシュヒット信号線３７７を介して与えられるヒット信号と、命令コード変換部３７０から信号線３７８を介して与えられる伸張完了を示すタイミング信号とを受け、出力がＲＥＡＤＹ信号線２７６に接続されたＯＲ回路３７５とを含む。命令コード変換部３７０は、キュー３７１と変換器３７２とを含む。

【００９５】

図２４を参照して、キャッシュメモリ３７４は１エントリが８バイトで容量が２ｋＢのダイレクトマップキャッシュである。キャッシュメモリ３５４の各エン

トリは、Ｖビット３８７およびＵビット３８２との２つのバリッドビットと、比較対照アドレスのタグ３８３と、命令コードであるデータ３８４とを含む。

【００９６】

図２２を参照して、アドレス線２７２から与えられたH’２０５０００－H’２００Ｃｆｆｆの一つを示す１９ビットのアドレスは、シフト回路２６１によって３ビット右にシフトされて１／８にアドレス変換され、その有効１６ビットがアドレス線２７４およびアドレス制御回路３７３を介してＪＡＶＡ命令のためのトランスレーション回路２６３および命令コード変換部３７０に与えられる。

【００９７】

図２４を参照して、キャッシュメモリ３７４では、このアドレスは上位８ビットのタグ３８５と下位８ビットのインデックス３８６とに分解される。インデックス３８６はエントリの選択に使用される。タグ３８５は各エントリから読み出されたタグ３８３との比較対照に用いられる。Ｖビット３８７が有効を示し、タグ３８３の値とタグ３８５の値とが等しいエントリがあれば、キャッシュヒットとなり、ヒット信号がアサートされる。一方当該エントリのデータ３８４が命令コードとして出力線３７９に出力される。図２３に示されるキャッシュヒット信号線３７７上に出力されるキャッシュヒット信号は、ＯＲ回路３７５を経てＲＥＡＤＹ信号線２７６上のＲＥＡＤＹ信号をアサートする。

【００９８】

再び図２３を参照して、命令コード変換部３７０では、キュー３７１が一旦ＪＡＶＡ命令をキューに入れた後、変換器３７２が１バイトずつＪＡＶＡ命令を読出し、ＶＬＩＷ命令に変換して命令コード変換部３７０の出力線３８０上に出力する。変換器３７２の処理クロック数はＪＡＶＡ命令の長さおよび種類、並びに変換後のＶＬＩＷ命令の数および種類に依存して変わる。

【００９９】

キャッシュヒット信号線３７７上のキャッシュヒット信号がアサートされた場合、ＭＵＸ３７６は出力線３７９を選択して、キャッシュメモリ３７４から出力された、変換後のＶＬＩＷ命令をデータ線２７９上に出力する。一方、キャッシュヒット信号がアサートされなかったとき、すなわちキャッシュメモリ３７４が

ミスしたときには、ＭＵＸ３７６は命令コード変換部３７０が命令コード変換部３７０の出力線３８０上に出力した変換後の８バイトのＶＬＩＷ命令を選択して、データ線２７９上に出力する。このとき、命令コード変換部３７０の出力線３８０上のＶＬＩＷ命令はキャッシュメモリ３７４にも転送され、対応するエントリに書込まれる。また、変換が完了し命令コード変換部３７０の出力線３８０上にＶＬＩＷ命令が出力されるタイミングで信号線３７８上のタイミング信号がアサートされ、このタイミング信号はＯＲ回路３７５ を経てＲＥＡＤＹ信号線２７６上のＲＥＡＤＹ信号をアサートする。

【０１００】

なお、ＪＡＶＡ命令を複数のＶＬＩＷ命令に変換する場合、命令コード変換部３７０はプロセッサ１０からのアクセスに関係なく変換後のＶＬＩＷ命令を全てキャッシュメモリ３７４に登録する。

【０１０１】

ＪＡＶＡ命令は、バイト単位で命令長が可変であり、メモリ上のどのアドレスにも配置可能である。そのため２バイト以上のＪＡＶＡ命令をメモリから読出す場合、ＪＡＶＡ命令のアドレス位置に依存してＪＡＶＡ命令用ＲＡＭ２６５を２回以上アクセスしなければならない可能性があることに注意が必要である。

【０１０２】

たとえば、２バイトのＪＡＶＡ命令Ｊ２が８バイト境界の最後のアドレスである（８ｎ＋７）番地にある場合を考える。このとき、ＪＡＶＡ命令用ＲＡＭ２６５では、１回目のアクセスで（８ｎ＋７）番地からＪ２の最後の１バイトを読出し、２回目のアクセスで８（ｎ＋１）番地からＪ２の次の１バイトを読出す必要がある。この２回目のアクセスのためのアドレス８（ｎ＋１）番地は、ＪＡＶＡ命令のためのトランスレーション回路２６３からアドレス線２７５に出力される。

【０１０３】

既に述べたように、変換器３７２によって変換されたＶＬＩＷ命令は、命令コード変換部３７０の出力線３８０を経てキャッシュメモリ３７４に１つずつ登録される。１つのＪＡＶＡ命令を複数のＶＬＩＷ命令に変換してキャッシュメモリ

３７４に登録する場合には、これら複数のＶＬＩＷ命令が常に組みになってキャッシュメモリ３７４に保持されているように配慮をする必要がある。そのために、先頭のＶＬＩＷ命令以外のエントリでは、Ｕビット３８２の「１」とする。こうすることにより、これらを含むエントリが、キャッシュメモリ３７４においてソフトウェアの関与なしに無効化されたり、キャッシュメモリ３７４から追い出されたりすることを禁止する。また、１つのＪＡＶＡ命令を複数のＶＬＩＷ命令に変換してキャッシュメモリ３７４に登録する場合、キャッシュが一杯で先頭のＶＬＩＷ命令以外のＶＬＩＷ命令がキャッシュメモリに登録できない場合もあり得る。この場合には、ＦＵＬＬ信号３８１をアサートし、このアサート信号に応答してソフトウェア制御によって先頭のＶＬＩＷ命令を含むエントリを無効化する。

【０１０４】

これは、１つのＪＡＶＡ命令を複数のＶＬＩＷ命令に変換した場合、たとえばＶＬＩＷ命令ＡとＶＬＩＷ命令Ｂとの間で割込が受付けられ、割込からの復帰後にＶＬＩＷ命令Ｂが実行されるときに、ＶＬＩＷ命令Ｂがキャッシュメモリにないという事態となって、１つのＪＡＶＡ命令の途中から命令変換が行なわれることになり間違った変換が行なわれてしまうことを防ぐためである。ＪＡＶＡ命令を含め、プロセッサの命令では一般に、途中のバイトに即値データがあり、命令の先頭バイトから見ない限り命令を正しく認識できないためである。なお、ＦＵＬＬ信号３８１がアサートされたときに、ソフトウェア制御によってＵビット３８２が「１」であるが対応する先頭のＶＬＩＷ命令がキャッシュにないために安全に無効化が可能なエントリがある場合、それらのエントリを無効化することにより、キャッシュメモリ３７４の空きエントリを増加させ、そこでＦＵＬＬ信号３８１をアサートするきっかけとなったＶＬＩＷ命令をキャッシュメモリ３７４に登録するようにしてもよい。

【０１０５】

さて、キャッシュメモリ３７４ において従来の通常のキャッシュメモリと同様の方法でエントリの無効化または新規登録のためのエントリの追出しを行なうと、割込がなくても変換器３７２で１つのＪＡＶＡ命令の途中から命令変換が行

なわれる可能性がある。これを防ぐために、本実施の形態の装置ではキャッシュメモリ３７４の各エントリにUビット３８２を設けた。１つのＪＡＶＡ命令から変換された、連続するアドレスにある複数の ＶＬＩＷ命令がキャッシュメモリ３７４に登録されたとき、先頭のＶＬＩＷ命令以外のＶＬＩＷ命令に対応するエントリではUビット３８２が「1」とされる。Uビット３８２が「1」のエントリは、ソフトウェアによる操作なしに無効化されたり、キャッシュメモリ３７４から追出されることはない。先にも述べたようにUビット３８２ が「1」のエントリの無効化または追出しは、そのエントリに含まれるＶＬＩＷ命令 とともにＪＡＶＡ命令から変換された複数の ＶＬＩＷ命令の先頭のＶＬＩＷ命令がキャッシュメモリ３７４内に存在しないときのみ可能である。これはソフトウェアにより管理される。

【０１０６】

これにより、１つのＪＡＶＡ命令から変換された複数のＶＬＩＷ命令の先頭のＶＬＩＷ命令がキャッシュメモリ３７４に登録されたままで、後続のＶＬＩＷ命令がキャッシュメモリ３７４に登録されてない状態が発生し、変換器３７２がＪＡＶＡ命令の途中から変換を開始することを防ぐ。

【０１０７】

図２５を参照して、命令コード変換部３７０が変換するＪＡＶＡ命令とＶＬＩＷ命令との関係は以下のとおりである。たとえば1バイトのＪＡＶＡ命令４０１は8バイトのＶＬＩＷ命令４１１に変換される。ＪＡＶＡ命令４０１は1バイトなので、このＪＡＶＡ命令４０１はＪＡＶＡ命令用ＲＡＭ２６５を1回アクセスするだけで読出すことができる。変換後のＶＬＩＷ命令４１１も8バイト境界に位置するため、キャッシュメモリ３７４の1エントリに登録される。

【０１０８】

２バイトのＪＡＶＡ命令４０２は2つの8バイトのＶＬＩＷ命令４１２ａ、４１２ｂに変換される。このとき、ＶＬＩＷ命令４１２ｂを含むエントリではUビット３８２が「1」となる。

【０１０９】

５バイトのＪＡＶＡ命令４０４は3つの8バイトのＶＬＩＷ命令４１４ａ、４

１４ｂおよび４１４ｃに変換される。ＶＬＩＷ命令４１４ｂおよび４１４ｃを含むエントリでは、Ｕビット３８２が共に「１」となる。

【０１１０】

ＪＡＶＡ命令４０３、ＪＡＶＡ命令４０６～４０８などについても同様で、ＪＡＶＡ命令４０３はＶＬＩＷ命令４１３に、ＪＡＶＡ命令４０６はＶＬＩＷ命令４１６に、ＪＡＶＡ命令４０７はＶＬＩＷ命令４１７に、ＪＡＶＡ命令４０８は３つの８バイトのＶＬＩＷ命令４１８ａ～４１８ｃに、それぞれ変換される。ＶＬＩＷ命令４１８ｂおよび４１８ｃを含むエントリでは、Ｕビット３８２が共に「１」となる。

【０１１１】

図２６～図３１は、変換器３７２でＪＡＶＡ命令をＶＬＩＷ命令に変換したときの具体的な例を示す。図２６を参照して、１バイトのＪＡＶＡ命令「iadd」はサブ命令「LDW R61，@(R63+，R0)」とサブ命令「ADD R62，R62，R61」とを順に実行する１つのＶＬＩＷ命令に変換される。「iadd」はスタックトップから１番目のデータと２番目のデータとである２つの３２ビット整数を加算してスタックに書き戻すＪＡＶＡ命令である。プロセッサ１０ではスタックトップのデータはレジスタＲ６２に配置され、レジスタＲ６３がスタックトップの次のデータのアドレスを示す。したがって、プロセッサ１０では、ＪＡＶＡ命令「iadd」のオペレーションを、スタックトップから２番目の３２ビットデータをレジスタＲ６１にロードしてレジスタＲ６２を４インクリメントするオペレーションを行なうサブ命令「LDW R61，@(R63+，R0)」と、レジスタＲ６２とレジスタＲ６１との２つの３２ビット整数を加算してその結果をレジスタＲ６２に書込むオペレーションを行なうサブ命令「ADD R62，R62，R61」とでエミュレートすることができる。このとき、ＰＣ値については、１つのＶＬＩＷ命令を実行したことによりプロセッサ１０のＰＣ値を８番地進めることで、ＪＡＶＡ命令「iadd」の実行に対応してＰＣ値を１進めることをエミュレートする。

【０１１２】

図２７を参照して、２バイトのＪＡＶＡ命令「iload」は、１つのサブ命令「ADD/CN R50，#(0||vindex)」を持つＶＬＩＷ命令と、サブ命令「STW R62，@(R63-

，R4)」とサブ命令「LDW R62，@(R10，R50)」とを順次に実行するＶＬＩＷ命令に変換される。「iload」はローカル変数領域から３２ビット整数をフェッチしてスタックトップに保存するＪＡＶＡ命令である。プロセッサ１０ではこれを、レジスタＲ５０にローカル変数のインデックス値をロードするサブ命令「ADD/CN R50，#(0||vindex)」と、レジスタＲ６２をスタックトップから２番目にプッシュするサブ命令「STW R62，@(R63-，R4)」と、スタックトップであるレジスタＲ６２にローカル変数領域からデータをロードするサブ命令「LDW R62，@(R10，R50)」とに分解することによりエミュレートする。

【０１１３】

なお、ここでレジスタＲ４には、値「－４」を保持し、レジスタＲ１０にはローカル変数領域のベースアドレスを保持する。ＰＣ値については２つのＶＬＩＷ命令を実行したことによりプロセッサ１０のＰＣ値を１６番地進めることで、ＪＡＶＡ命令「iload」の実行に対応してＰＣ値を２番地進めることをエミュレートする。

【０１１４】

図２８を参照して、３バイトのＪＡＶＡ命令「ifeq」は、サブ命令「ADD R62，R61，R0」とサブ命令「NOP」とを並列に実行するＶＬＩＷ命令と、サブ命令「LDW R62，@(R63+，R0)」とサブ命令「NOP」とを並列に実行するＶＬＩＷ命令と、１つのサブ命令「BRATZR/CN R62，#(s||branchbyte1||branchbyte2)」を実行するＶＬＩＷ命令とに分解される。命令「ifeq」はスタックトップのデータが「０」なら分岐するＪＡＶＡ命令である。プロセッサ１０ではこれを、レジスタＲ６２をレジスタＲ６１にコピーするサブ命令「ADD R62，R61，R0」と、スタックトップから２番目のデータをレジスタＲ６２にポップするサブ命令「LDW R62，@(R63+，R0)」と、レジスタＲ６１がゼロなら分岐するサブ命令「BRATZR/CN R62，#(s||branchbyte1||branchbyte2)」とに分解し、これら３つのサブ命令と２つのNOP命令とを組合せて、計３つのＶＬＩＷ命令でエミュレートする。ＰＣ値については、３つのＶＬＩＷ命令を実行したことによりプロセッサ１０のＰＣ値を２４番地進めることで、ＪＡＶＡ命令「ifeq」の実行に対応してＰＣ値を３番地進めることをエミュレートする。

【０１１５】

図２９を参照して、５バイトのＪＡＶＡ命令「jsr_w」は、１つのサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」からなるＶＬＩＷ命令と、サブ命令「STW R62, @(R63-, R4)」とサブ命令「BSR R10」とを順次に実行するＶＬＩＷ命令と、サブ命令「BRA #3」とサブ命令「NOP」とを並列に実行するＶＬＩＷ命令とに変換される。「jsr_w」は戻り先アドレスをスタックにプッシュして４バイトで指定されたアドレスのサブルーチンへジャンプするＪＡＶＡ命令である。プロセッサ１０ではこの命令を、ジャンプ先アドレスをレジスタＲ１０にロードするサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」と、スタックトップのレジスタＲ６２の値をスタックトップから２番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタＲ６２に保存してレジスタＲ１０で指定されたアドレスのサブルーチンへジャンプするサブ命令「JSR R10」と、サブルーチンから復帰した後に２つのＶＬＩＷ命令をスキップするための分岐を行なうサブ命令「BRA #3」と、サブ命令「NOP」とに分解してエミュレートする。ＰＣ値については、３つのＶＬＩＷ命令を実行し、２つのＶＬＩＷ命令をスキップする分岐を行なうことによりプロセッサ１０のＰＣ値を４０番地進めることで、ＪＡＶＡ命令「jsr_w」の実行に対応してＰＣ値を５番地進めることをエミュレートする。

【０１１６】

図３０と図３１とは、複雑なＪＡＶＡ命令を、ＶＬＩＷ命令からなりかつＪＡＶＡ命令の機能を実行するサブルーチンをコールするＶＬＩＷ命令に変換する例を示す。図３０を参照して、この例では、浮動小数点数の加算を行なうＪＡＶＡ命令「fadd」を、スタックトップのレジスタＲ６２の値をスタックトップから２番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタＲ６２に保存して#faddで指定されたアドレスのサブルーチンにジャンプするサブ命令「JSR #fadd」とを順次に実行するＶＬＩＷ命令に変換する。プロセッサ１０では、「fadd」のオペレーションをサブルーチン中で行ない、ＰＣ値の更新に関しては、一つのＶＬＩＷ命令を実行することで

プロセッサ１０のＰＣ値を８番地進めて「fadd」に対応するＰＣ値を１番地進めることをエミュレートする。

【０１１７】

図３１に示す例では、テーブルジャンプを行なうＪＡＶＡ命令「tableswitch」を、スタックトップのレジスタＲ６２の値をスタックトップから２番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタＲ６２に保存して#tableswitchで指定されたアドレスのサブルーチンへジャンプするサブ命令「JSR #tableswitch」とを順次に実行するＶＬＩＷ命令に変換する。プロセッサ１０では、「tableswitch」のオペレーションとＰＣ値の更新との両方をサブルーチン中でＪＡＶＡ命令「tableswitch」で指定された各種パラメータをアクセスしてエミュレートする。このとき、プロセッサ１０は、ＪＡＶＡ命令用（直結）アドレス領域１２５ａでＪＡＶＡ命令用のトランスレータ付メモリ２５をアクセスしてＪＡＶＡ命令「tableswitch」で指定された各種パラメータをデータとして読出す。

【０１１８】

図３２を参照して、プロセッサ１０がＪＡＶＡ命令用のトランスレータ付メモリ２５を使用してＪＡＶＡ命令で記述されたプログラムをエミュレートする時の、プロセッサ１０で実行される処理のアルゴリズムは、以下のような制御構造を有する。まずプロセッサ１０は、ＲＯＭ２３からＪＡＶＡプログラムをＪＡＶＡ命令用のトランスレータ付メモリ２５ にロードする（４２０）。次にプロセッサ１０は、ＪＡＶＡの実行環境を作り、エミュレーションのための初期設定を行なう（４２１）。この後、プロセッサ１０はＪＡＶＡ命令用のトランスレータ付メモリ２５のアドレスにジャンプすることによりエミュレーションを開始する（４２２）。

【０１１９】

キャッシュメモリ３７４をアクセスして、ヒットならば制御はステップ４２８に進み、ミスならば制御はステップ４２４に進む（４２３）。キャッシュミスの場合、ＪＡＶＡ命令用ＲＡＭ２６５からＪＡＶＡ命令をフェッチし、命令コード変換部３７０でＪＡＶＡ命令をネイティブのＶＬＩＷ命令 に変換する（４２４

）。キャッシュメモリ３７４に変換後のネイティブ命令を登録可能ならばステップ４２７に進む。キャッシュメモリ３７４の対応するエントリがフルで登録不可の場合にはＦＵＬＬ信号３８１をアサートしてプロセッサ１０に対して割込を要求し、ソフトウェア制御でキャッシュメモリ３７４に変換後のネイティブ命令を登録するための空きエントリを作成する（４２５、４２６）。

【０１２０】

次に、プロセッサ１０で変換後のネイティブ命令を実行してＪＡＶＡ命令をエミュレートする（４２８）。エミュレーションが終了でないならばＰＣ値を進めてキャッシュメモリ３７４をアクセスする。エミュレーションが終了であればＪＡＶＡ命令用のトランスレータ付メモリ２５の動作を終了してネイティブ命令のプログラムに戻る。

【０１２１】

ここで、ステップ４２０～４２２、４２６の処理は、ＪＡＶＡ命令を変換したネイティブ命令での処理ではなく、ネイティブ命令用ＲＡＭ２１に存在するネイティブのＶＬＩＷ命令のプログラムを実行することにより行なう。

【０１２２】

非ネイティブ命令Ｘ用のトランスレータ付メモリ２６ は、８バイトの固定長の命令Ｘをプロセッサ１０のＶＬＩＷ命令に変換して実行するためのものである。プロセッサ１０から非ネイティブ命令Ｘ用のトランスレータ付メモリ２６を見た場合、そのアドレスは図１１に示す非ネイティブ命令Ｘ（直結）用アドレス領域１２６ａと非ネイティブ命令Ｘ（トランスレータ経由）用アドレス領域１２６ｂとからなる。どちらのアドレス領域で非ネイティブ命令Ｘ用のトランスレータ付メモリ２６をアクセスしても６４ｋＢの空間の全バイト位置に有効なメモリが存在する。アドレス領域１２６ｂで非ネイティブ命令Ｘ用のトランスレータ付メモリ２６をアクセスする場合、命令Ｘがトランスレータ１６でＶＬＩＷ命令に変換されて出力される。トランスレータ１６の詳細は図１２に示されるトランスレータ１４とほぼ同様である。ただし、トランスレータ１６においては、図１２に示されるアドレス変換器２４１、ＭＵＸ２４２およびアドレス制御回路３７３に対応する回路を持たず、アドレスが変換されずに入力されること、キャッシュメ

モリにＵビット３８２とＦＵＬＬ信号３８１とに対応する機能がないこと、および常に１つの命令Ｘが１つのＶＬＩＷ命令に変換されることがトランスレータ１４と異なる。

【０１２３】

以上の実施の形態の装置では、キャッシュメモリの１エントリに１つのＶＬＩＷ命令を保持する場合について説明した。しかしこの発明はそうした構成には限定されず、１エントリに複数のＶＬＩＷ命令を保持するようにしてもよい。また、上記実施の形態の装置ではＪＡＶＡ命令の１バイトを１つのＶＬＩＷ命令に対応させてエミュレートしているが、ＪＡＶＡ命令の１バイトについて２または３のＶＬＩＷ命令命令を対応させてエミュレートするようにしてもよい。

【０１２４】

さらに、上記実施の形態では変換後の命令をキャッシュに保持する例を述べた。しかしこの発明はそうした構成に限定されるわけではなく、このキャッシュメモリはなくてもよい。また、キャッシュではなく変換後の複数のＶＬＩＷ命令を一時的に保持し、使用後は毎回保持内容を無効化するバッファメモリを用いてもよい。

【０１２５】

今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【０１２６】

【発明の効果】

以上のように請求項１に記載の発明によれば、プロセッサ本体の構成を変更することなく非ネイティブ命令をネイティブ命令に変換してプロセッサで実行することができ、かつ高速に変換後の命令を出力できるので、効率的に非ネイティブ命令で記述されたプログラムのエミュレートを行なうことができる。

【０１２７】

請求項２に記載の発明によれば、請求項１に記載の発明の効果に加えて、非ネ

イティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がない。そのため非ネイティブ命令で記述されたプログラムのエミュレートが容易に行える。

【０１２８】

請求項３に記載の発明によれば、請求項２に記載の発明の効果に加えて、非ネイティブ命令のプログラムカウンタの値を明示的にエミュレートする必要がなくエミュレートがさらに容易に行え、かつプログラムを保存するためのメモリが小さくてすみ、ハードウェアのコストの増大が回避できる。

【０１２９】

請求項４に記載の発明によれば、請求項３に記載の発明の効果に加えて、非ネイティブ命令である第２の命令体系の命令のエミュレートを容易に行なうことができる。さらに、非ネイティブ命令の長さも第１の命令体系のネイティブ命令の長さでエミュレートでき、非ネイティブ命令で記述されたプログラムのエミュレートがさらに容易になる。

【０１３０】

請求項５に記載の発明によれば、請求項１に記載の発明の効果に加えて、命令の変換効率がよく、効率的に非ネイティブ命令で記述されたプログラムのエミュレートを行なうことができる。

【０１３１】

請求項６に記載の発明によれば、請求項１に記載の発明の効果に加えて、命令の変換が容易であり、かつ、非ネイティブ命令のプログラムカウンタ値を変換後のネイティブ命令の長さでエミュレートすることができるので、容易に、かつ効率的に、非ネイティブ命令で記述されたプログラムのエミュレートを行なうことができる。

【０１３２】

請求項７に記載の発明によれば、請求項６に記載の発明の効果に加えて、非ネイティブ命令とネイティブ命令のサブ命令とを対照させることにより、変換が容易に行える。そのため、非ネイティブ命令で記述されたプログラムのエミュレートをさらに容易に行なうことができる。

【０１３３】

請求項８に記載の発明によれば、変換後のネイティブ命令を高速にプロセッサに与えることができる。さらに、複数のネイティブ命令の同時無効化など、無効化のための条件が複雑な場合でも容易に対処できる。その結果、非ネイティブ命令で記述されたプログラムのエミュレートが高速かつ確実に行える。

【０１３４】

請求項９に記載の発明によれば、請求項８に記載の発明の効果に加えて、ソフトウェアの責任において安全に保持手段の保持内容を維持することができるので、非ネイティブ命令で記述されたプログラムのエミュレートをより確実に行える。

【０１３５】

請求項１０に記載の発明によれば、請求項８に記載の発明の効果に加えて、無効化しても安全なエントリを無効化することをソフトウェア処理によって明示的に指示して、新たな命令を保持可能とすることができるので、保持手段の保持内容を安全かつ確実に維持することができる。その結果、非ネイティブ命令で記述されたプログラムのエミュレートをより確実にかつ高速に行える。

【０１３６】

請求項１１に記載の発明によれば、プロセッサの変更なしに、非ネイティブ命令により記述されたプログラムをプロセッサのネイティブ命令で実行できる。

【０１３７】

請求項１２に記載の発明によれば、請求項１１に記載の発明の効果に加えて、第２の命令体系で記述されたプログラムを他のメモリに転送したり、その内容を解析したりすることが可能になり、プログラム開発および保守が容易に行える。

【０１３８】

請求項１３に記載の発明によれば、請求項１１に記載の発明の効果に加えて、命令変換時と、それ以外のときとで、トランスレータ付命令メモリに関して異なるメモリマップを使用することができ、トランスレータ付命令メモリの用途に応じて適切にアクセスすることが可能になる。

【０１３９】

請求項１４に記載の発明によれば、請求項１１に記載の発明の効果に加えて、高速に変換後の命令を出力することができるので、非ネイティブ命令により記述されたプログラムを高速にエミュレートすることができる。

【０１４０】

請求項１５に記載の発明によれば、プロセッサの構成を変更することなく、第２の命令体系の命令で記述されたプログラムをこのプロセッサで実行することが可能となり、ハードウェアの増大を招くことなく異なる命令体系の命令で記述されたプログラムをエミュレートできる。

【０１４１】

請求項１６に記載の発明によれば、請求項１５に記載の発明の効果に加えて、プロセッサ本体の変更なしに非ネイティブ命令もネイティブ命令も区別なくプロセッサでデコードし実行することができ、非ネイティブ命令で記述されたプログラムのエミュレートがより容易になる。

【０１４２】

請求項１７に記載の発明によれば、請求項１６に記載の発明の効果に加えて、プロセッサでは命令が非ネイティブ命令の場合もネイティブ命令の場合も区別なくフェッチしてデコードすることができ、非ネイティブ命令で記述されたプログラムのエミュレートがより容易になる。

【０１４３】

請求項１８に記載の発明によれば、請求項１６に記載の発明の効果に加えて、複数種類の非ネイティブ命令で記述されたプログラムを、区別なくネイティブ命令を用いて実行でき、より幅広い範囲の非ネイティブ命令で記述されたプログラムのエミュレートを容易に行なうことができる。

【図面の簡単な説明】

【図１】　従来の旧命令エミュレーション機能付データ処理装置の概略ブロック図である。

【図２】　本発明の１実施の形態にかかる命令トランスレータ機能付メモリを備えたデータ処理装置のブロック図である。

【図３】　図２に示すプロセッサ１０のブロック図である。

【図４】　プロセッサ１０が有するレジスタの一覧を表形式で示す図である。

【図５】　プロセッサ１０の制御レジスタ１７０の詳細を示す図である。

【図６】　プロセッサ１０のパイプライン処理機構を説明するための図である。

【図７】　プロセッサ１０で実行可能なＶＬＩＷ命令のフォーマットを示す図である。

【図８】　プロセッサ１０で実行可能なＶＬＩＷ命令のサブ命令のフォーマットを示す図である。

【図９】　プロセッサ１０のサブ命令のパイプライン処理方法を示すための図である

【図１０】　プロセッサ１０で２つのサブ命令をパイプライン処理する方法を説明するための図である。

【図１１】　図２に示すデータ処理装置のメモリマップを示す図である。

【図１２】　図２のデータ処理装置内の、圧縮命令を伸張する圧縮命令用のトランスレータ付メモリ２４の詳細を示す図である。

【図１３】　圧縮命令を変換するためのトランスレーション回路２４３の詳細を示す図である。

【図１４】　図１３に示すキャッシュメモリ３５４の詳細を示す図である。

【図１５】　図２に示すデータ処理装置のバスサイクルのタイミングチャートである。

【図１６】　２つの圧縮命令をＶＬＩＷ命令に伸張するときのビットフィールドの対応を示す図である。

【図１７】　圧縮命令を伸張する具体例を示す図である。

【図１８】　圧縮命令を伸張する具体例を示す図である。

【図１９】　圧縮命令を伸張する具体例を示す図である。

【図２０】　圧縮命令を伸張する具体例を示す図である。

【図２１】　本実施の形態のデータ処理装置において、圧縮命令でアクセスできる汎用レジスタの一連を示す図である。

【図２２】　図２に示した、ＪＡＶＡ命令を変換するＪＡＶＡ命令用のトランスレータ付メモリ２５の詳細を示す図である。

【図２３】　ＪＡＶＡ命令のためのトランスレーション回路２６３の詳細を示す図である。

【図２４】　ソフトウェアでしか無効化できないエントリを備えたキャッシュメモリ３７４の詳細を示す図である。

【図２５】　ＪＡＶＡ命令を１または複数のＶＬＩＷ命令に変換するときの命令間の対応を示す図である。

【図２６】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図２７】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図２８】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図２９】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図３０】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図３１】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図３２】　図２に示すデータ処理装置がＪＡＶＡ命令のエミュレーションを行なう際の処理の流れを示すフローチャートである。

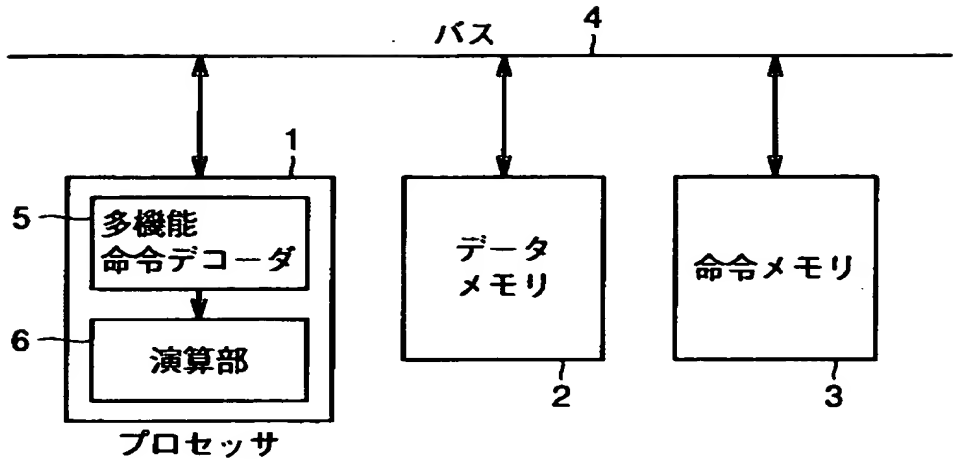
【符号の説明】

１０　プロセッサ、　２０　バス制御回路、　４０　バス、１４、１５、１６　命令トランスレータ、２４　圧縮命令用のトランスレータ付メモリ、２１　ネイティブ命令用ＲＡＭ、２２　データ用ＲＡＭ、２３　ＲＯＭ、２５　ＪＡＶＡ命令用のトランスレータ付メモリ、２６　非ネイティブ命令Ｘ用トランスレータ付メモリ、２４３、２６３　トランスレーション回路、２４５　圧縮命令用ＲＡＭ、２６５　ＪＡＶＡ命令用ＲＡＭ、３５４、３７４　キャッシュメモリ、３５

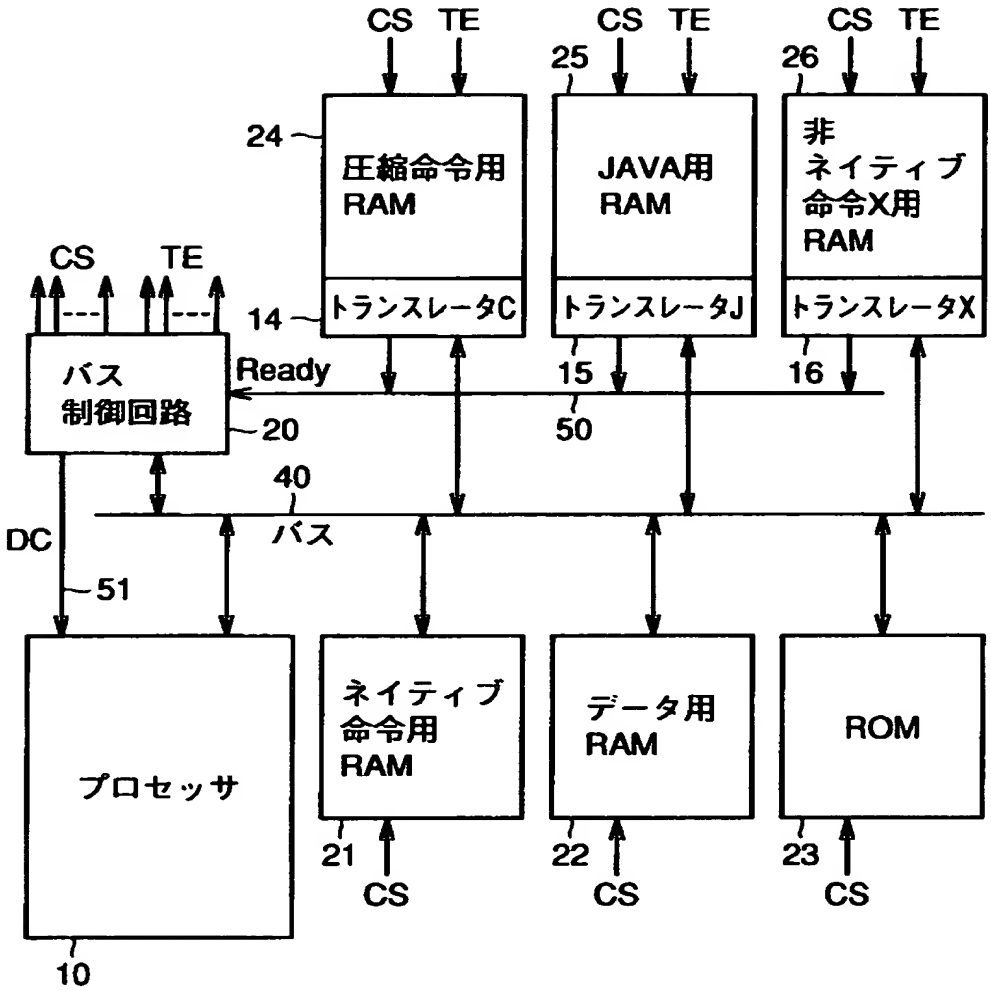
０、３７０　命令コード伸張部。

【書類名】　　図面

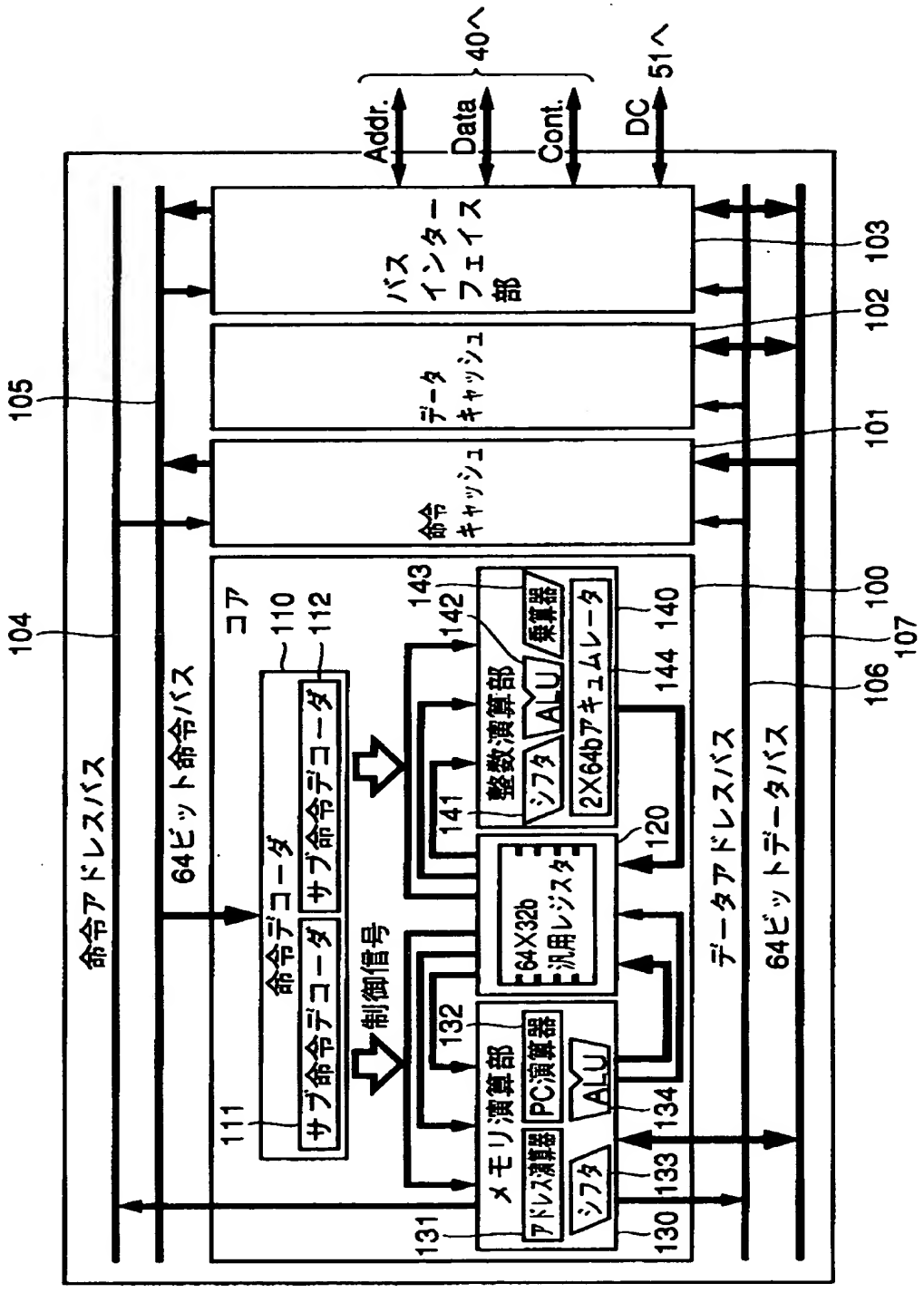
【図１】



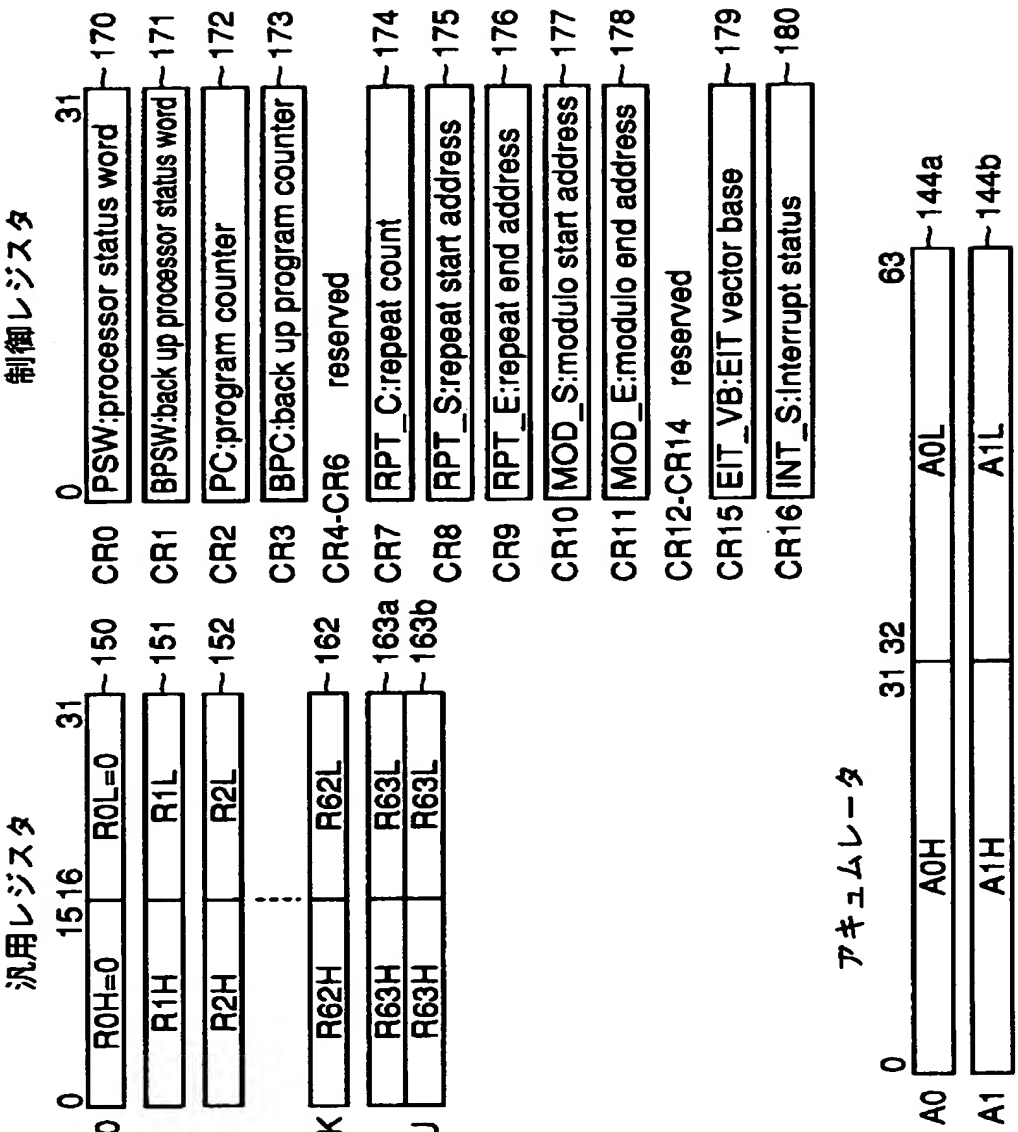
【図２】


CS
TE
CS
TE
CS
TE
25
26
24
圧縮命令用
RAM
JAVA用
RAM
非
ネイティブ
命令X用
RAM
CS
TE
14
トランスレータC
トランスレータJ
トランスレータX
バス
制御回路
Ready
15
16
20
50
40
バス
DC
51
プロセッサ
ネイティブ
命令用
RAM
データ用
RAM
ROM
21
22
23
CS
CS
CS
10

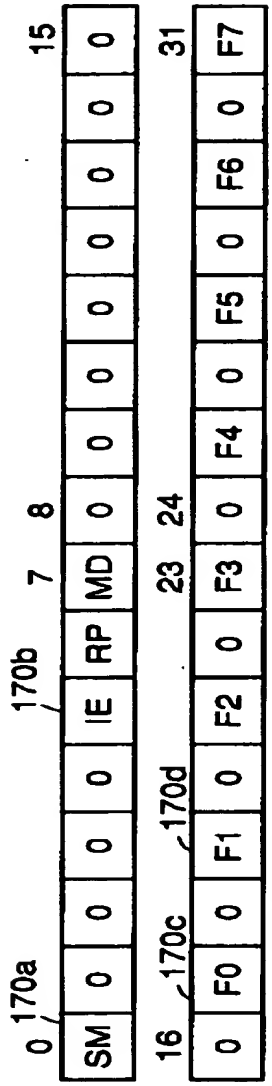
【図３】


Addr.
Data
Cont.
DC
40へ
51へ
バス インターフェイス 部
データ キャッシュ
命令 キャッシュ
103
102
101
100
105
104
107
106
命令アドレスバス
64ビット命令バス
データアドレスバス
64ビットデータバス
コア
110
112
命令デコーダ
サブ命令デコーダ
命令デコーダ
サブ命令デコーダ
制御信号
143
142
141
整数演算部
シフタ
ALU
乗算器
2X64bアキュムレータ
140
144
120
64X32b 汎用レジスタ
132
メモリ演算部
PC演算器
ALU
シフタ
アドレス演算器
111
131
133
134
130

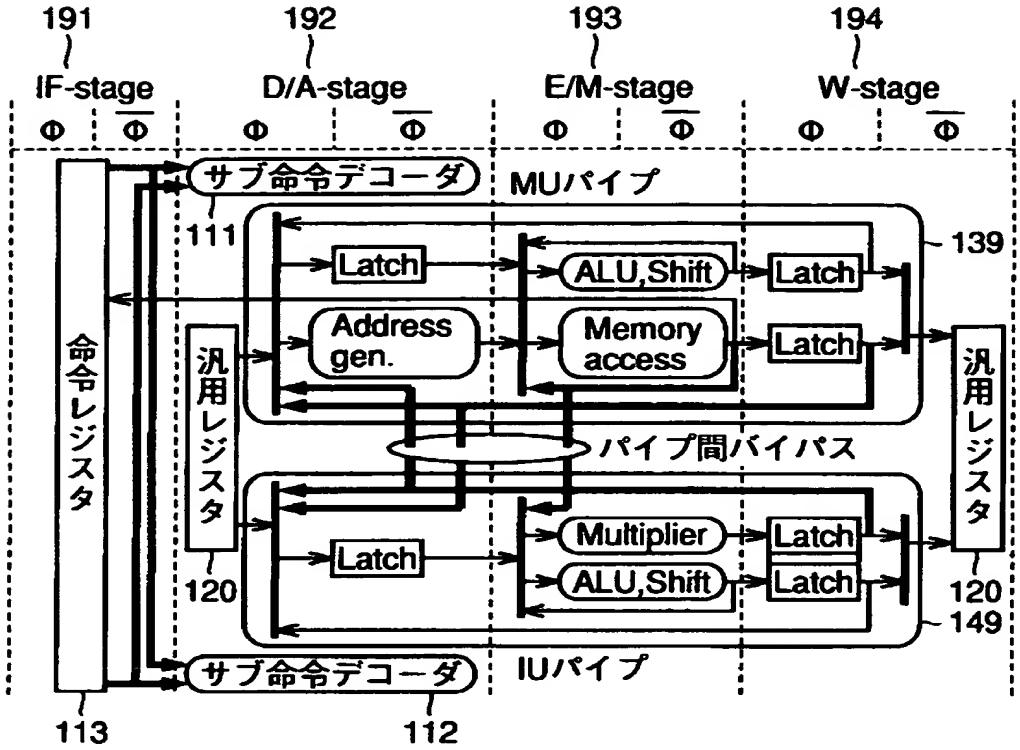
【図４】


汎用レジスタ
制御レジスタ
アキュムレータ
R0=0
R1
R2
R62=LINK
R63=SPI
R63=SPU
R0H=0
R0L=0
R1H
R1L
R2H
R2L
R62H
R62L
R63H
R63L
CR0
CR1
CR2
CR3
CR4-CR6
CR7
CR8
CR9
CR10
CR11
CR12-CR14
CR15
CR16
PSW:processor status word
BPSW:back up processor status word
PC:program counter
BPC:back up program counter
reserved
RPT_C:repeat count
RPT_S:repeat start address
RPT_E:repeat end address
MOD_S:modulo start address
MOD_E:modulo end address
reserved
EIT_VB:EIT vector base
INT_S:Interrupt status
A0
A1
A0H
A0L
A1H
A1L

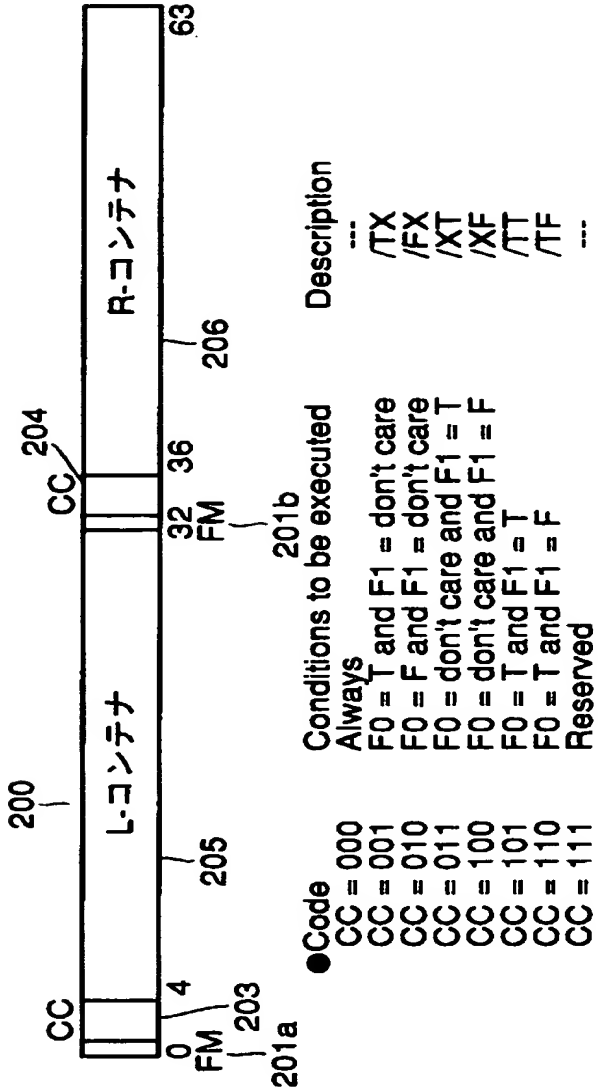
【図５】


0
170a
SM
0
0
0
0
IE
RP
MD
170b
7
8
15
0
0
0
0
0
0
0
0
16
170c
F0
0
F1
0
F2
0
F3
0
170d
23
24
31
F4
0
F5
0
F6
0
F7
● SM:Stack mode
SM=0 Interrupt mode.SPI is used.
SM=1 User mode.SPU is used.
● IE:Interrupt enable
IE=0 Interrupts are masked.
IE=1 Interrupts are accepted.
● RP:Repeat enable
RP=0 A block repeat is inactive.
RP=1 A block repeat is active.
● MD:Modulo enable
MD=0 Modulo addressing is disabled.
MD=1 Modulo addressing is enabled.
● F0 :Execution control flag #0
● F1 :Execution control flag #1
● F2 :General flag
● F3 :General flag
● F4(S) :Saturation flag
● F5(V) :Overflow flag
● F6(VA) :Accumulated Overflow flag
VA is cleared by a reset interrupt and MVTSYS instruction.
● F7(C) :Carry/Borrow flag

【図６】

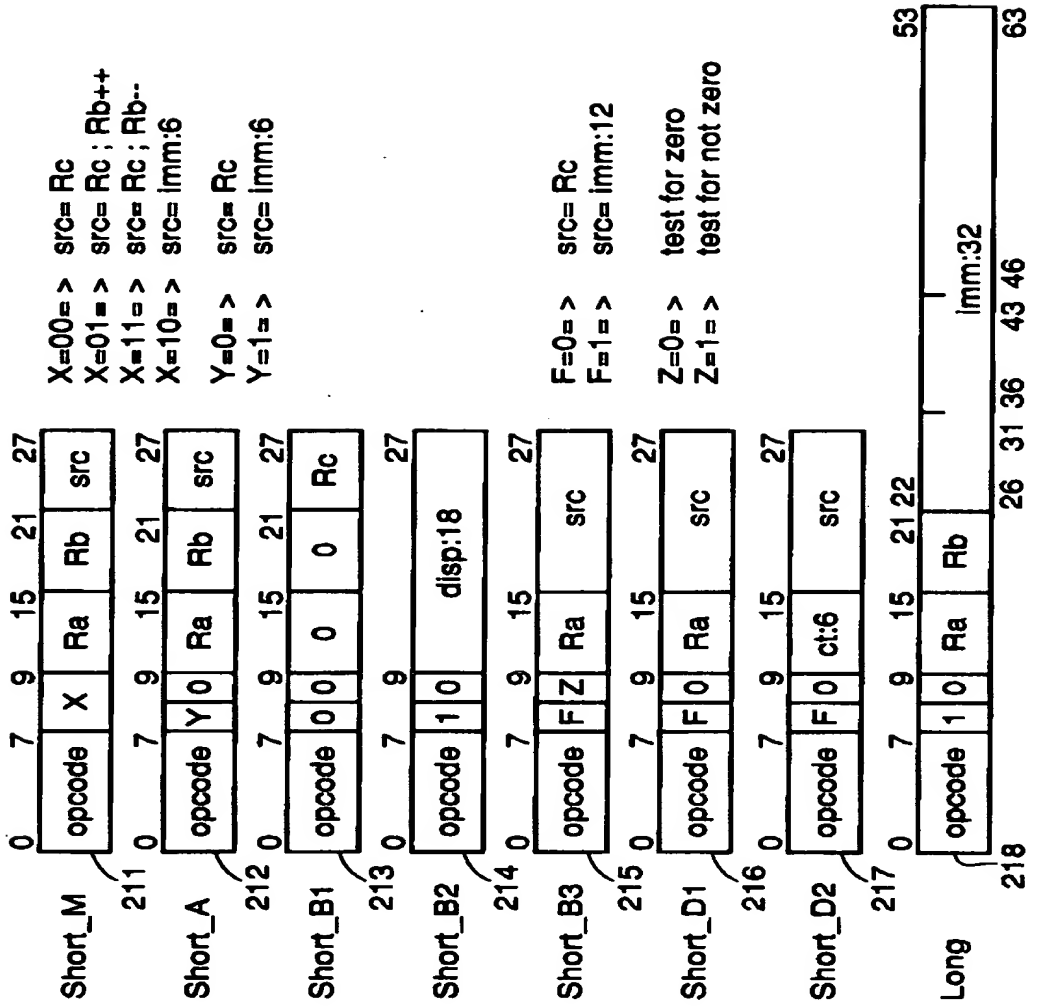

191
192
193
194
IF-stage
D/A-stage
E/M-stage
W-stage
Φ
Φ
Φ
Φ
Φ
Φ
Φ
Φ
サブ命令デコーダ
MUパイプ
111
139
Latch
ALU,Shift
Latch
Address gen.
Memory access
Latch
命令レジスタ
汎用レジスタ
汎用レジスタ
パイプ間バイパス
Multiplier
Latch
Latch
ALU,Shift
Latch
120
120
149
IUパイプ
サブ命令デコーダ
113
112

【図７】

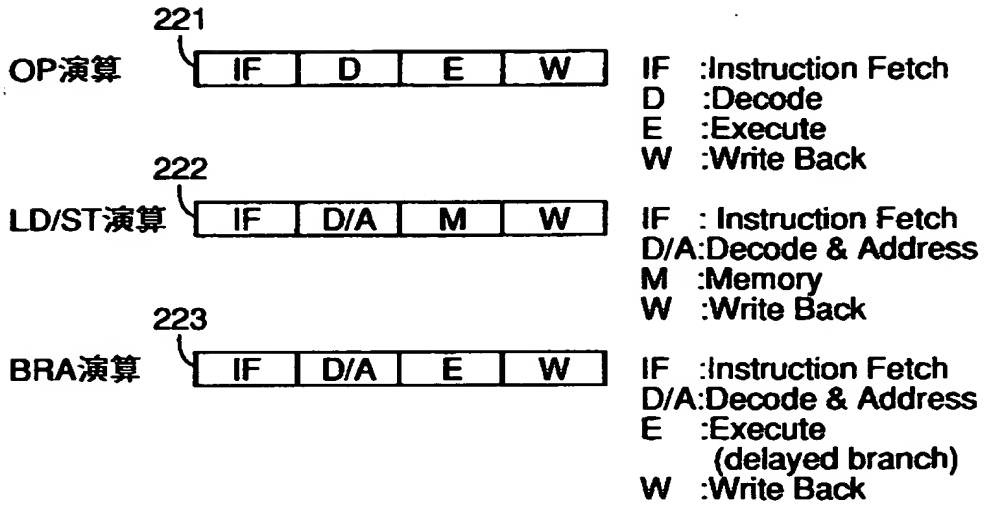


FM	Number of sub-instructions	Issuing Order L-container	Issuing Order R-container
00	two	1st	1st
01	two	1st	2nd
10	two	2nd	1st
11	one	1st	----

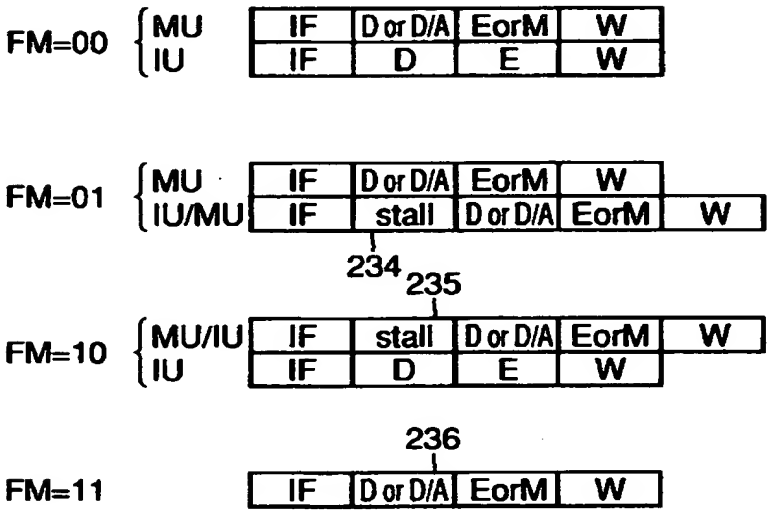
【図８】


X=00= > src= Rc
X=01= > src= Rc ; Rb++
X=11= > src= Rc ; Rb--
X=10= > src= imm:6
Y=0= > src= Rc
Y=1= > src= imm:6
F=0= > src= Rc
F=1= > src= imm:12
Z=0= > test for zero
Z=1= > test for not zero
Short_M
211
Short_A
212
Short_B1
213
Short_B2
214
Short_B3
215
Short_D1
216
Short_D2
217
Long
218
opcode
X
Y
F
Z
Ra
Rb
src
Rc
disp:18
ct:6
imm:32
0
7
9
15
21
22
26
27
31
36
43
46
53
63

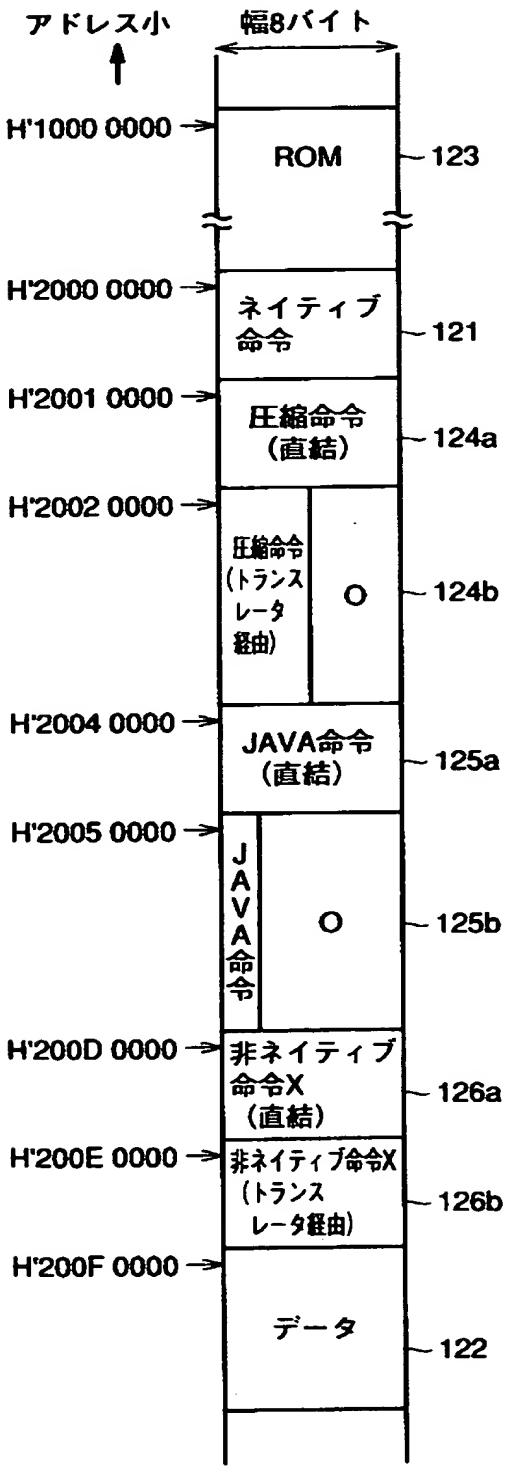
【図９】


221
OP演算
IF
D
E
W
IF :Instruction Fetch
D :Decode
E :Execute
W :Write Back
222
LD/ST演算
IF
D/A
M
W
IF : Instruction Fetch
D/A:Decode & Address
M :Memory
W :Write Back
223
BRA演算
IF
D/A
E
W
IF :Instruction Fetch
D/A:Decode & Address
E :Execute
(delayed branch)
W :Write Back

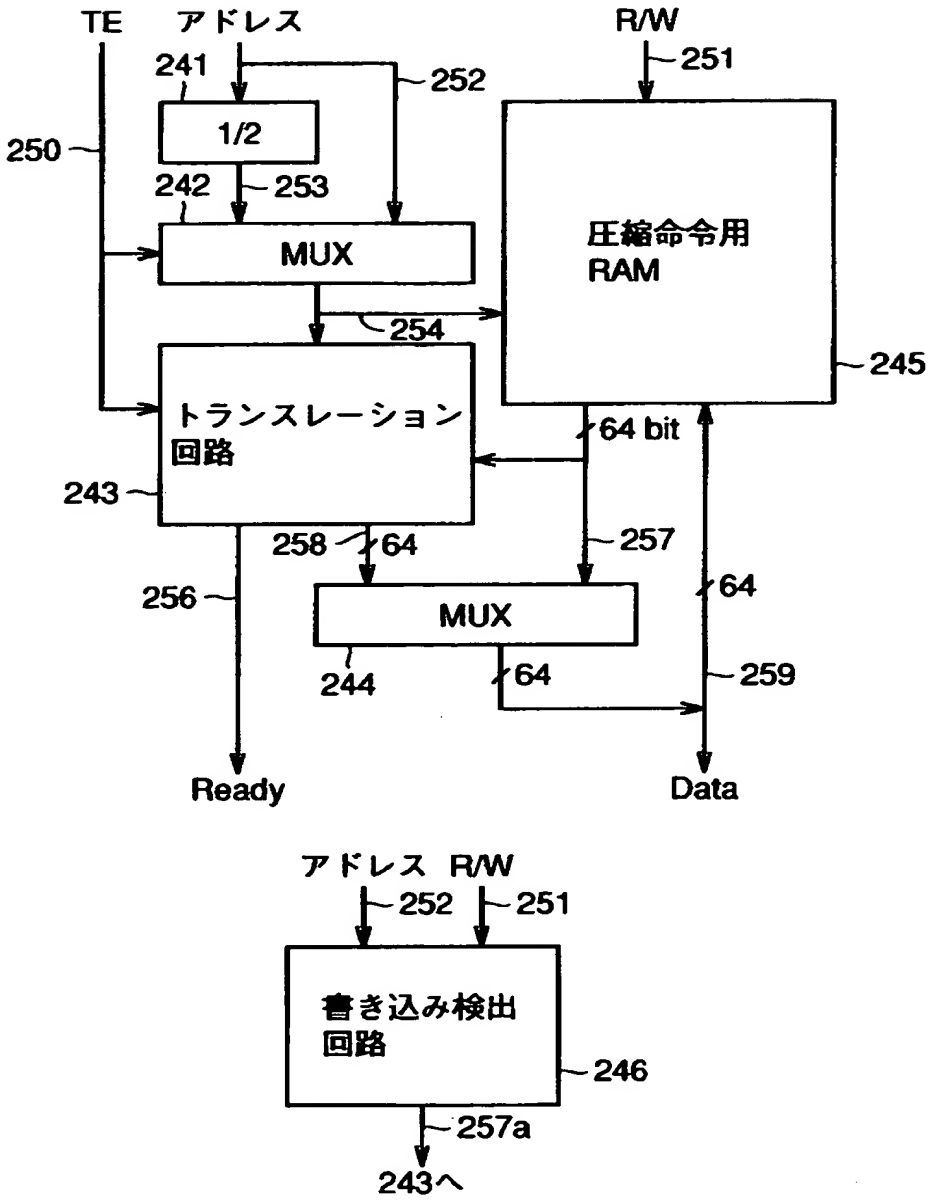
【図１０】


FM=00
MU
IU
IF
D or D/A
EorM
W
IF
D
E
W
FM=01
MU
IU/MU
IF
D or D/A
EorM
W
IF
stall
D or D/A
EorM
W
234
235
FM=10
MU/IU
IU
IF
stall
D or D/A
EorM
W
IF
D
E
W
236
FM=11
IF
D or D/A
EorM
W

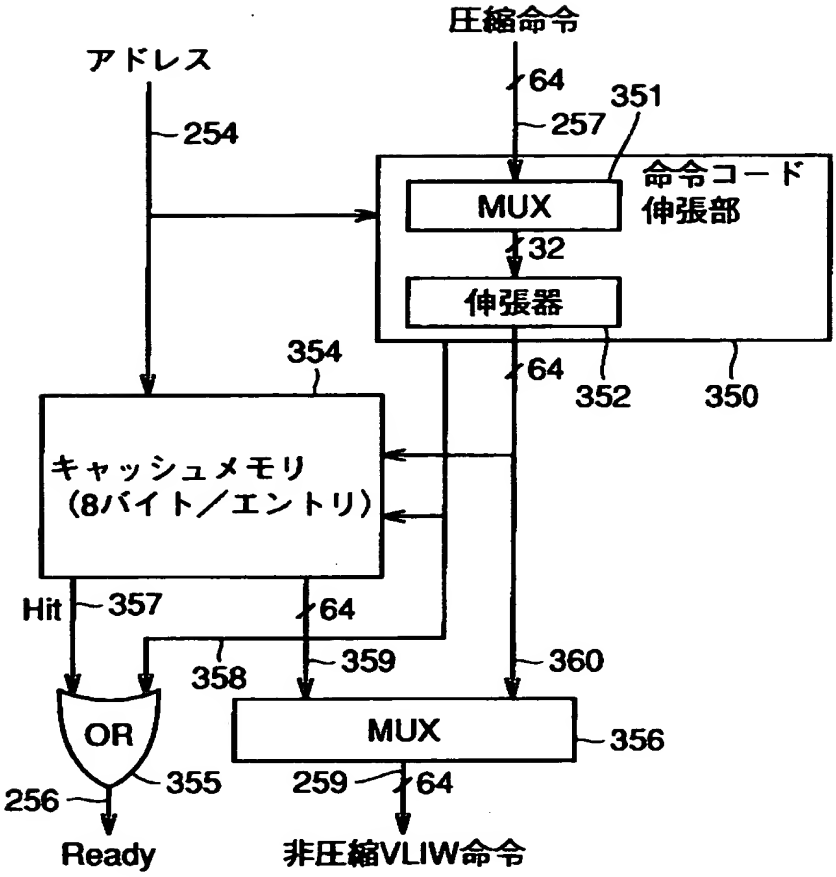
【図１１】


アドレス小
幅8バイト
H'1000 0000
ROM
123
H'2000 0000
ネイティブ命令
121
H'2001 0000
圧縮命令（直結）
124a
H'2002 0000
圧縮命令（トランスレータ経由）
O
124b
H'2004 0000
JAVA命令（直結）
125a
H'2005 0000
JAVA命令
O
125b
H'200D 0000
非ネイティブ命令X（直結）
126a
H'200E 0000
非ネイティブ命令X（トランスレータ経由）
126b
H'200F 0000
データ
122

【図１２】

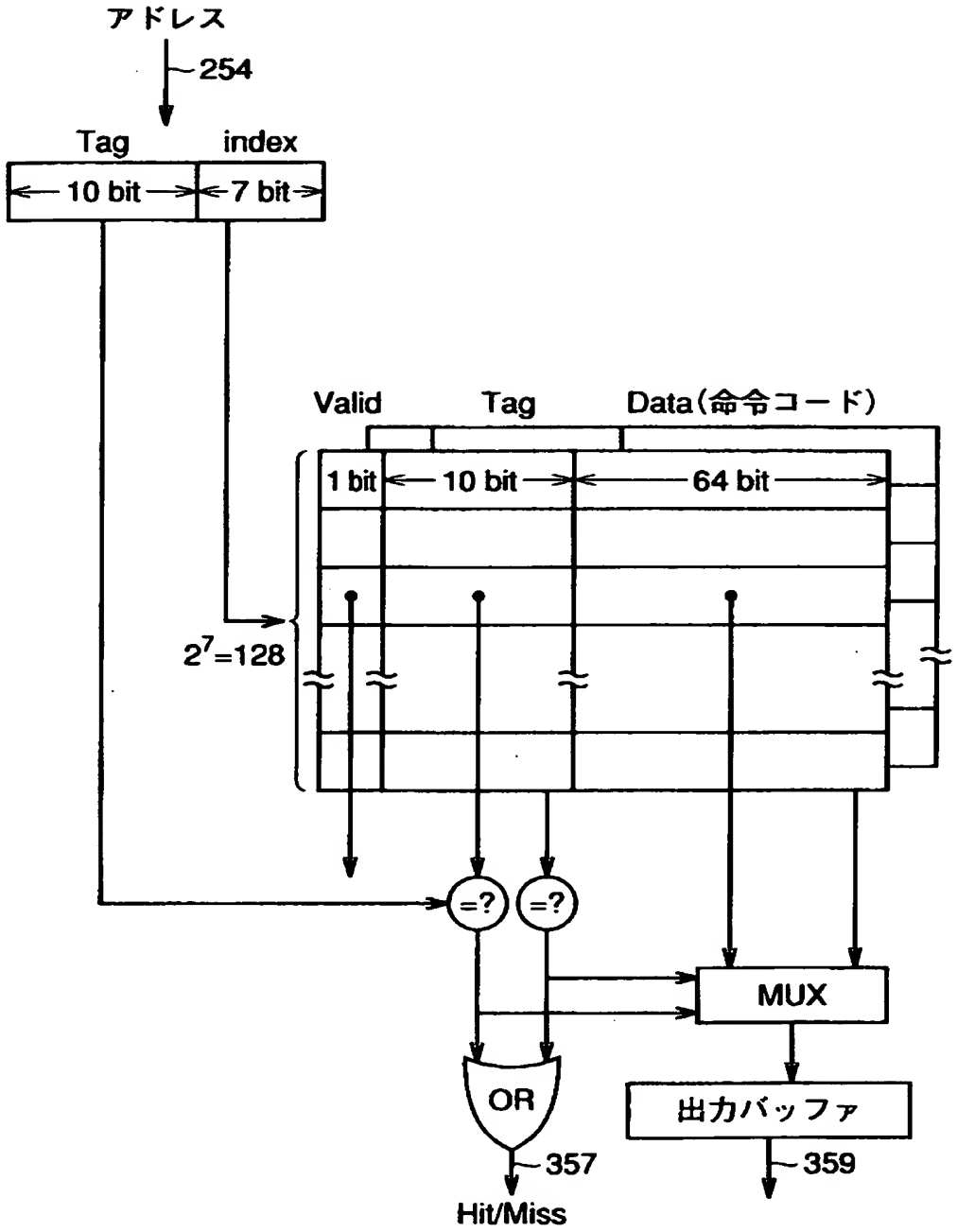

TE
アドレス
R/W
241
252
251
1/2
250
242
253
MUX
圧縮命令用
RAM
254
245
トランスレーション
回路
64 bit
243
258
64
257
256
MUX
64
244
64
259
Ready
Data
アドレス
R/W
252
251
書き込み検出
回路
246
257a
243へ

【図１３】

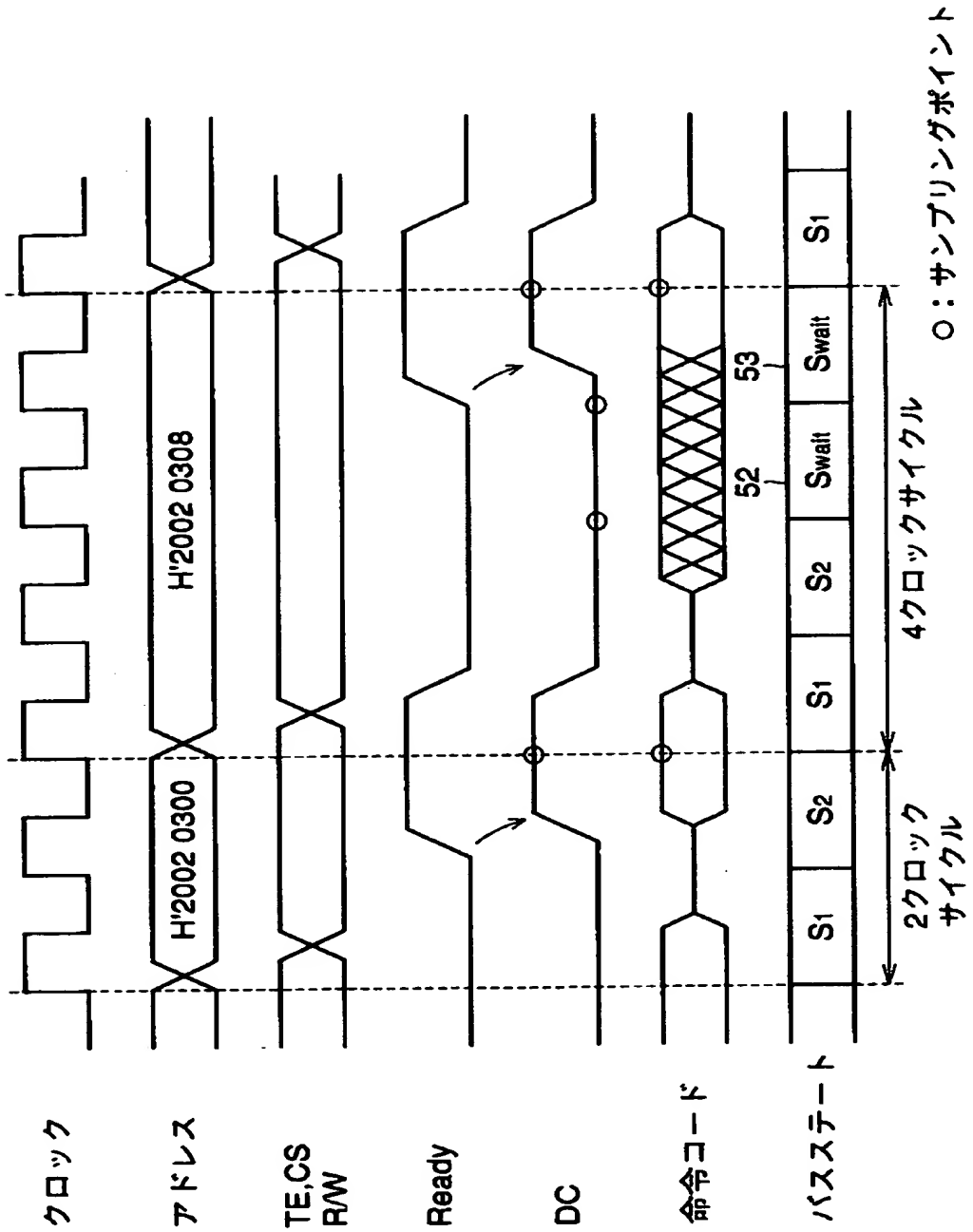


トランスレーション回路

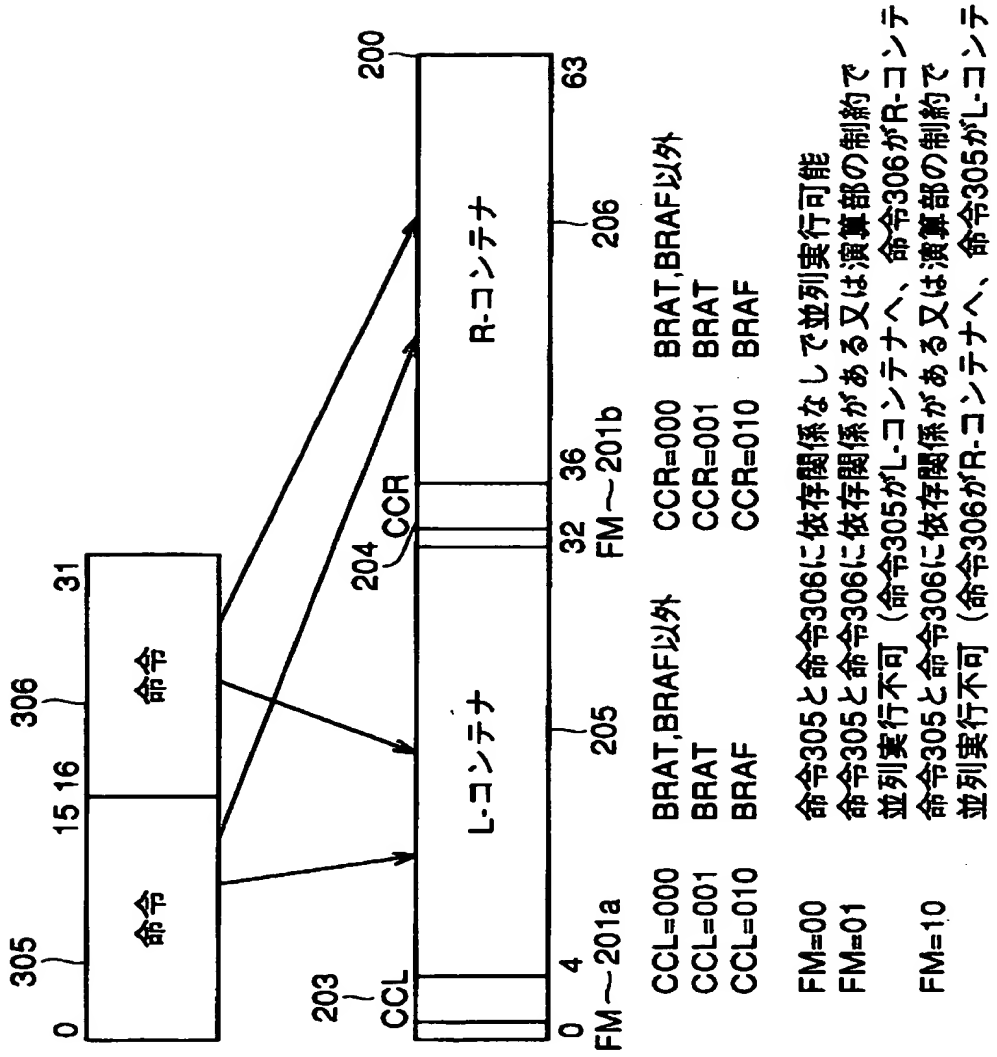
【図１４】


アドレス
254
Tag
index
10 bit
7 bit
Valid
Tag
Data(命令コード)
1 bit
10 bit
64 bit
2^7=128
=?
=?
MUX
OR
出力バッファ
357
359
Hit/Miss

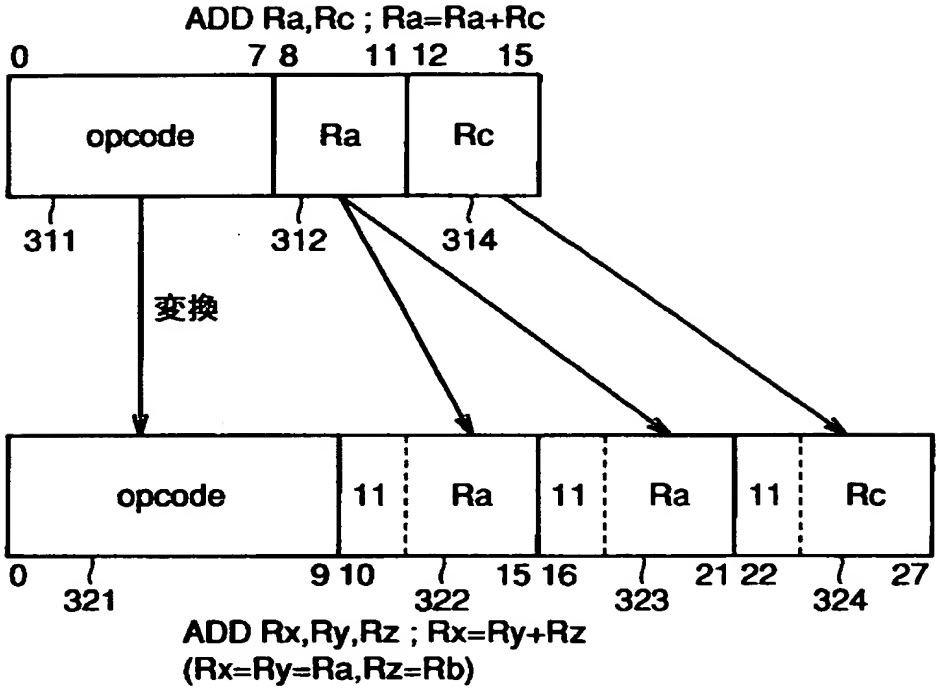
【図１５】


クロック
アドレス
TE,CS
R/W
Ready
DC
命令コード
バスステート
H'2002 0308
H'2002 0300
S1
S2
S1
S2
Swait
Swait
S1
52
53
○：サンプリングポイント
4クロックサイクル
2クロック
サイクル

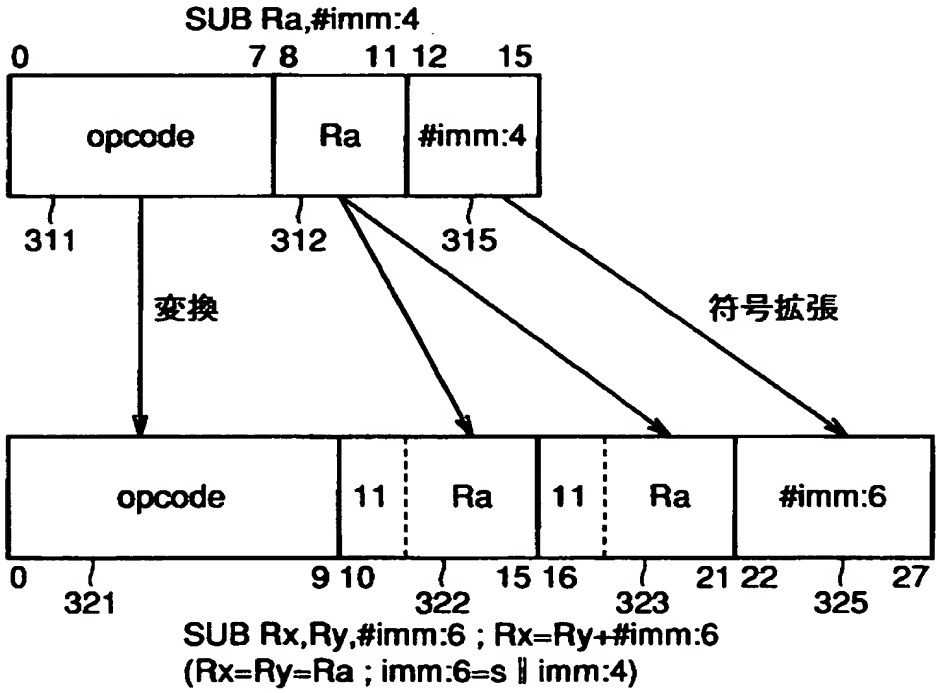
【図１６】


命令
命令
305
306
0
15 16
31
L-コンテナ
R-コンテナ
200
63
206
205
204
CCR
32 36
FM～201b
203
CCL
0 4
FM～201a
CCL=000 BRAT,BRAF以外
CCL=001 BRAT
CCL=010 BRAF
CCR=000 BRAT,BRAF以外
CCR=001 BRAT
CCR=010 BRAF
FM=00 命令305と命令306に依存関係なしで並列実行可能
FM=01 命令305と命令306に依存関係がある又は演算部の制約で
並列実行不可（命令305がL-コンテナへ、命令306がR-コンテナへ）
FM=10 命令305と命令306に依存関係がある又は演算部の制約で
並列実行不可（命令306がR-コンテナへ、命令305がL-コンテナへ）

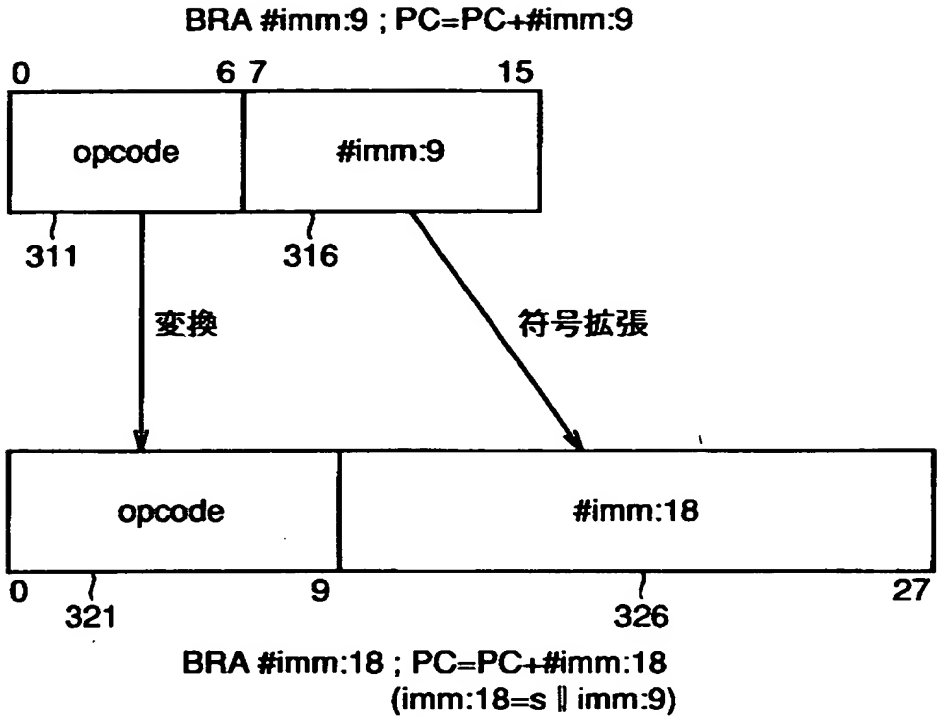
【図１７】


ADD Ra,Rc ; Ra=Ra+Rc
0
7 8
11 12
15
opcode
Ra
Rc
311
312
314
変換
opcode
11
Ra
11
Ra
11
Rc
0
9 10
15 16
21 22
27
321
322
323
324
ADD Rx,Ry,Rz ; Rx=Ry+Rz
(Rx=Ry=Ra,Rz=Rb)

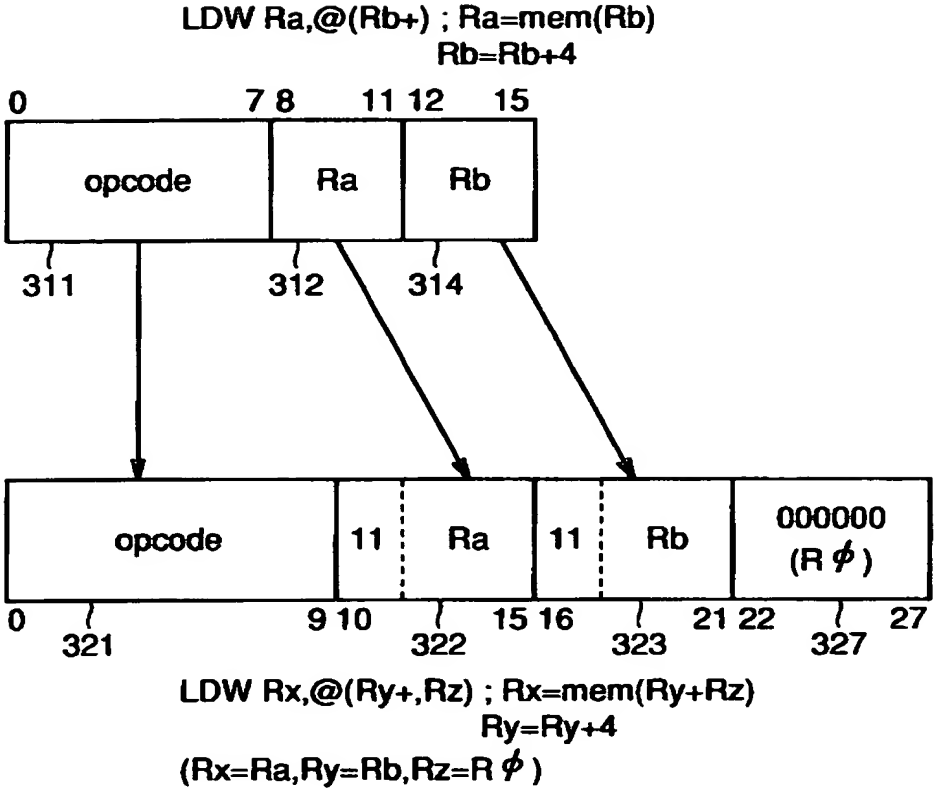
【図１８】


SUB Ra,#imm:4
0
7 8
11 12
15
opcode
Ra
#imm:4
311
312
315
変換
符号拡張
opcode
11
Ra
11
Ra
#imm:6
0
9 10
15 16
21 22
27
321
322
323
325
SUB Rx,Ry,#imm:6 ; Rx=Ry+#imm:6
(Rx=Ry=Ra ; imm:6=s ‖ imm:4)

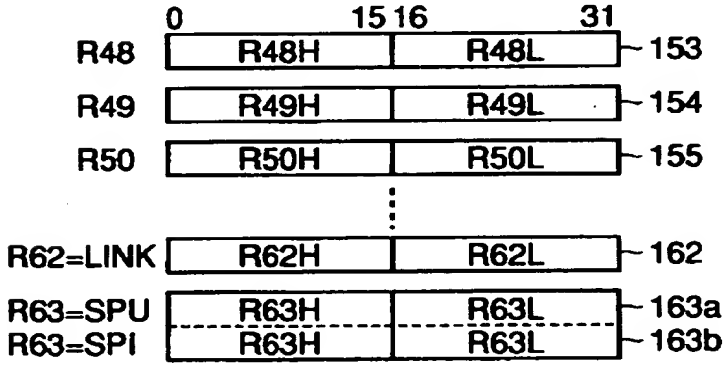
【図１９】


BRA #imm:9 ; PC=PC+#imm:9
0
6 7
15
opcode
#imm:9
311
316
変換
符号拡張
opcode
#imm:18
0
321
9
326
27
BRA #imm:18 ; PC=PC+#imm:18
(imm:18=s ‖ imm:9)

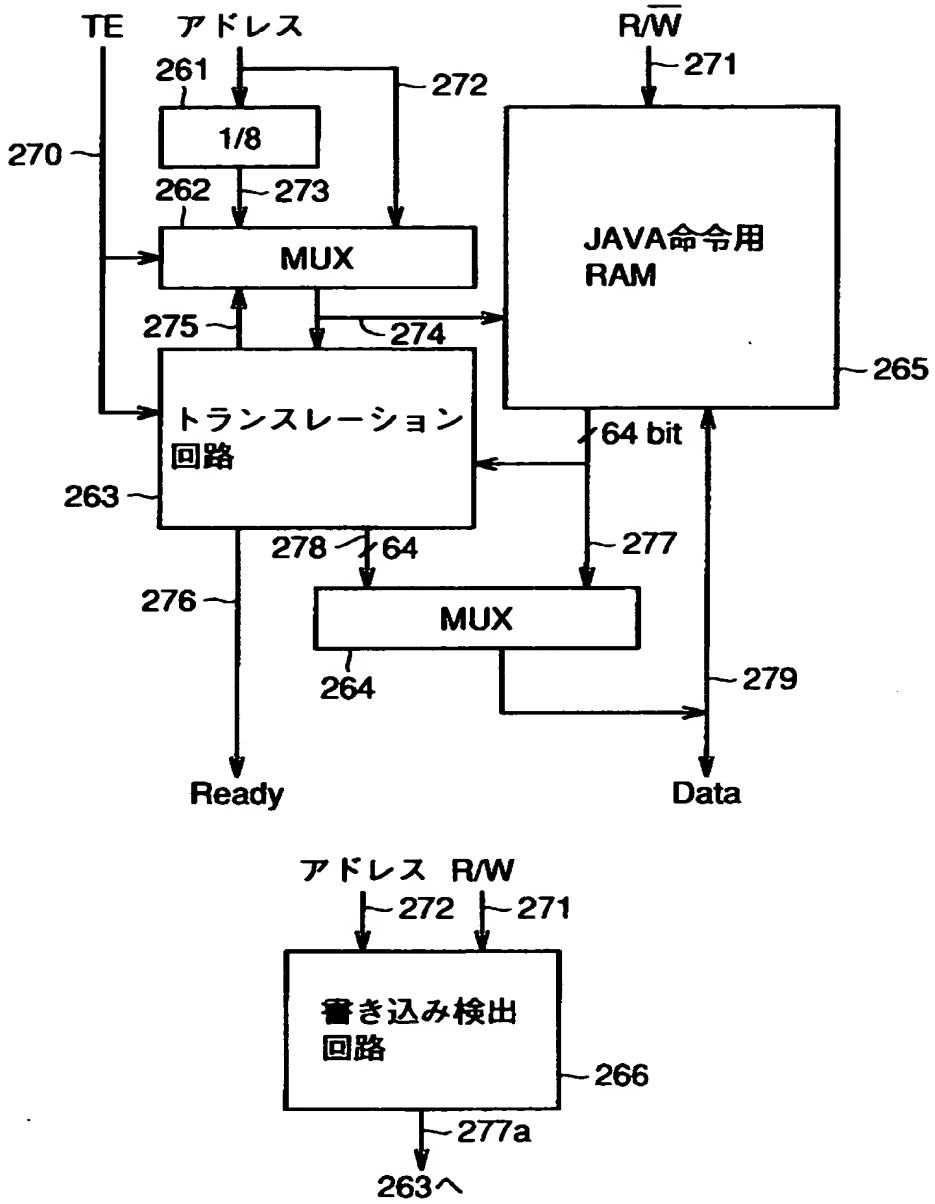
【図２０】


LDW Ra,@(Rb+) ; Ra=mem(Rb)
Rb=Rb+4
0
7 8
11 12
15
opcode
Ra
Rb
311
312
314
opcode
11
Ra
11
Rb
000000
(Rφ)
0
9 10
15 16
21 22
27
321
322
323
327
LDW Rx,@(Ry+,Rz) ; Rx=mem(Ry+Rz)
Ry=Ry+4
(Rx=Ra,Ry=Rb,Rz=Rφ)

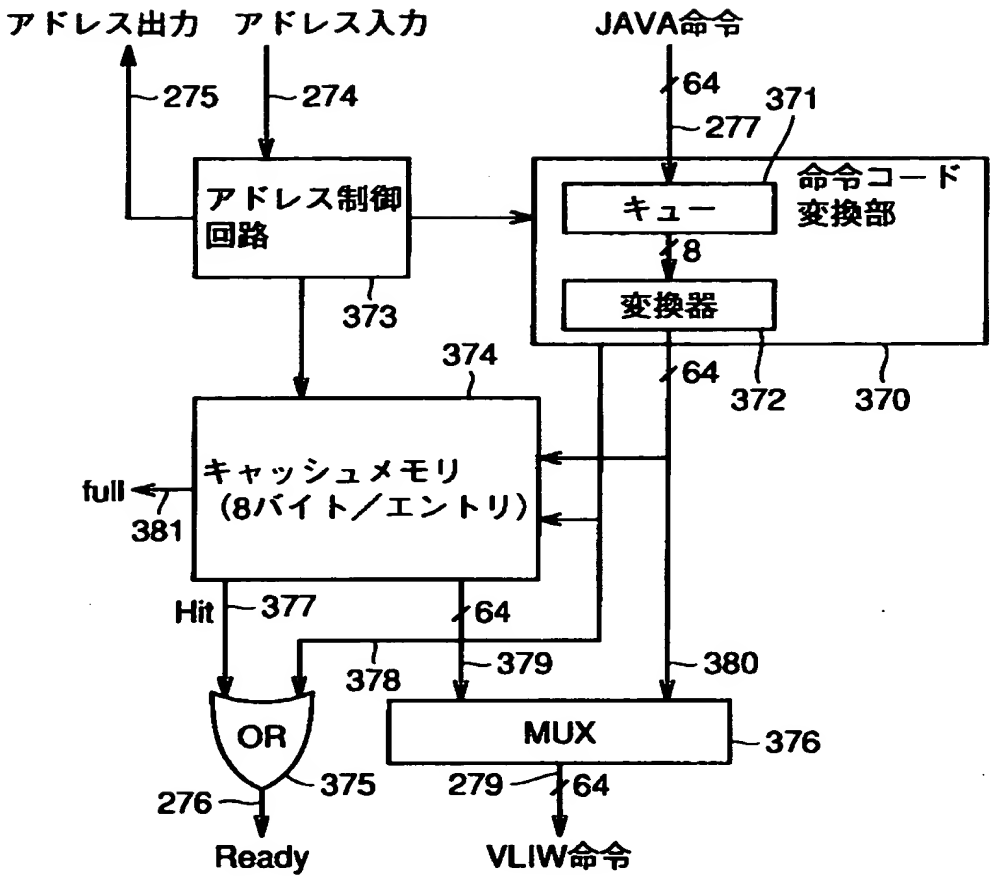
【図２１】


0
15 16
31
R48
R48H
R48L
153
R49
R49H
R49L
154
R50
R50H
R50L
155
R62=LINK
R62H
R62L
162
R63=SPU
R63H
R63L
163a
R63=SPI
R63H
R63L
163b

【図２２】

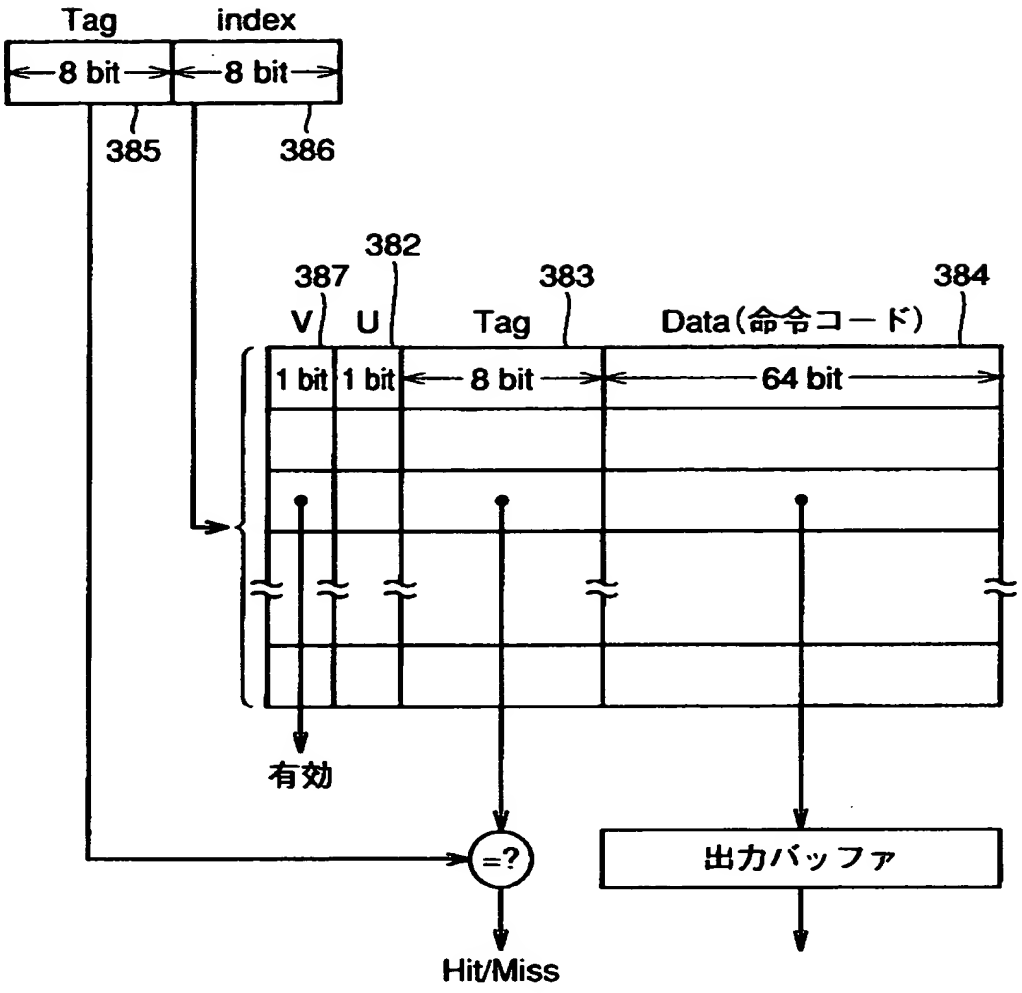

TE
アドレス
261
272
1/8
270
262
273
MUX
275
274
トランスレーション
回路
263
278
64
276
MUX
264
R/W
271
JAVA命令用
RAM
265
64 bit
277
279
Ready
Data
アドレス
R/W
272
271
書き込み検出
回路
266
277a
263へ

【図２３】

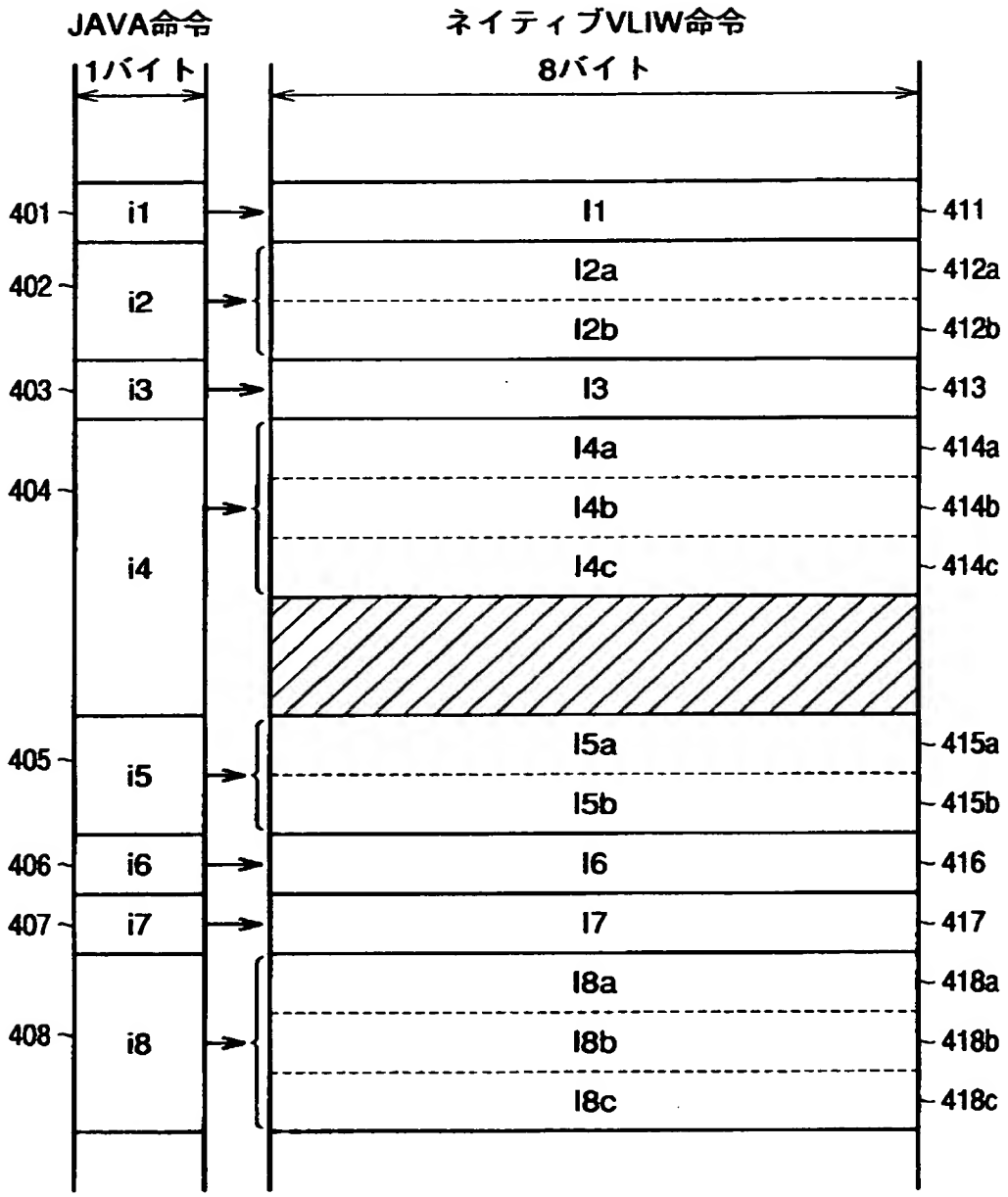


トランスレーション回路

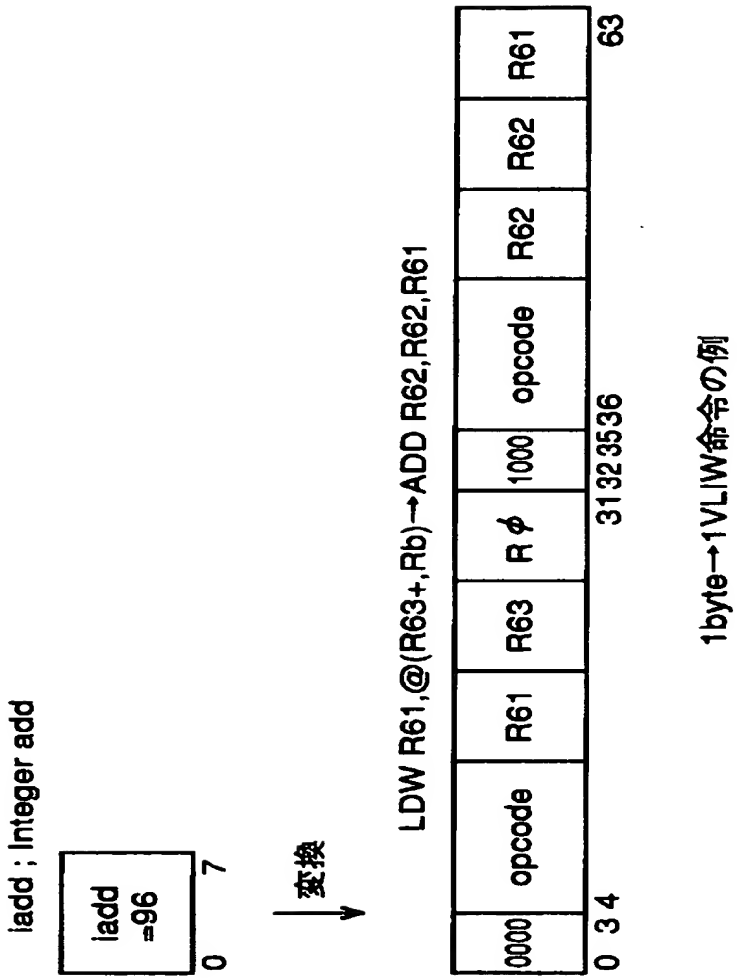
【図２４】


Tag
index
8 bit
8 bit
385
386
382
387
383
384
V
U
Tag
Data(命令コード)
1 bit
1 bit
8 bit
64 bit
有効
=?
出力バッファ
Hit/Miss

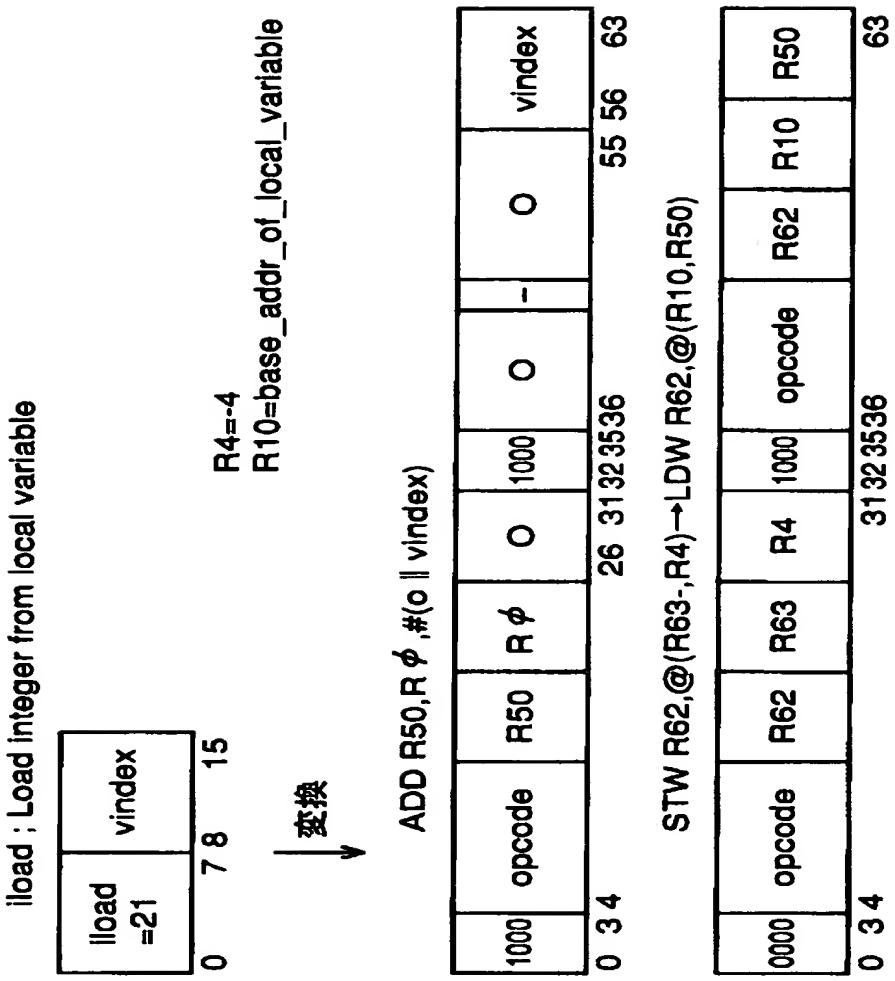
【図２５】


JAVA命令
1バイト
ネイティブVLIW命令
8バイト
401
i1
I1
411
402
i2
I2a
412a
I2b
412b
403
i3
I3
413
404
i4
I4a
414a
I4b
414b
I4c
414c
405
i5
I5a
415a
I5b
415b
406
i6
I6
416
407
i7
I7
417
408
i8
I8a
418a
I8b
418b
I8c
418c

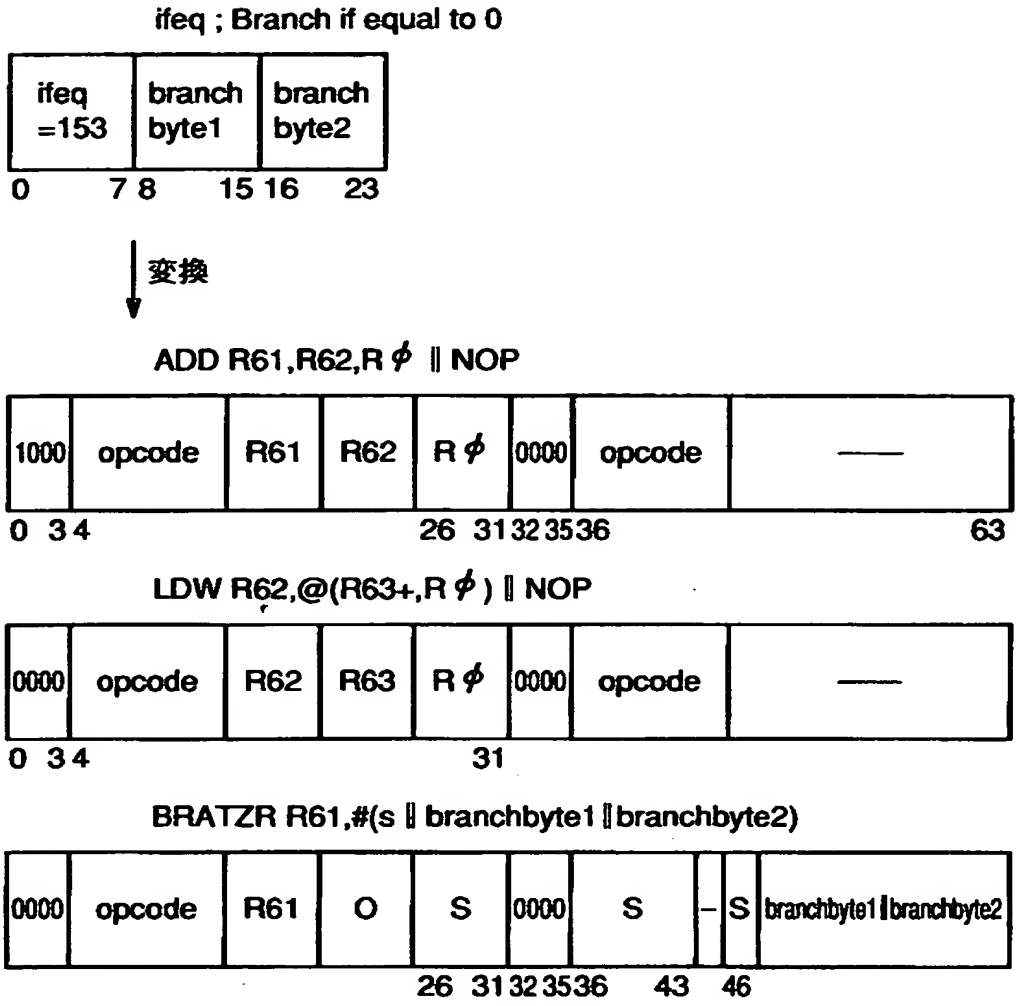
【図２６】


iadd ; Integer add
iadd
=96
0
7
変換
LDW R61,@(R63+,Rb)→ADD R62,R62,R61
0000
opcode
R61
R63
R φ
1000
opcode
R62
R62
R61
0 3 4
31 32 35 36
63
1byte→1VLIW命令の例

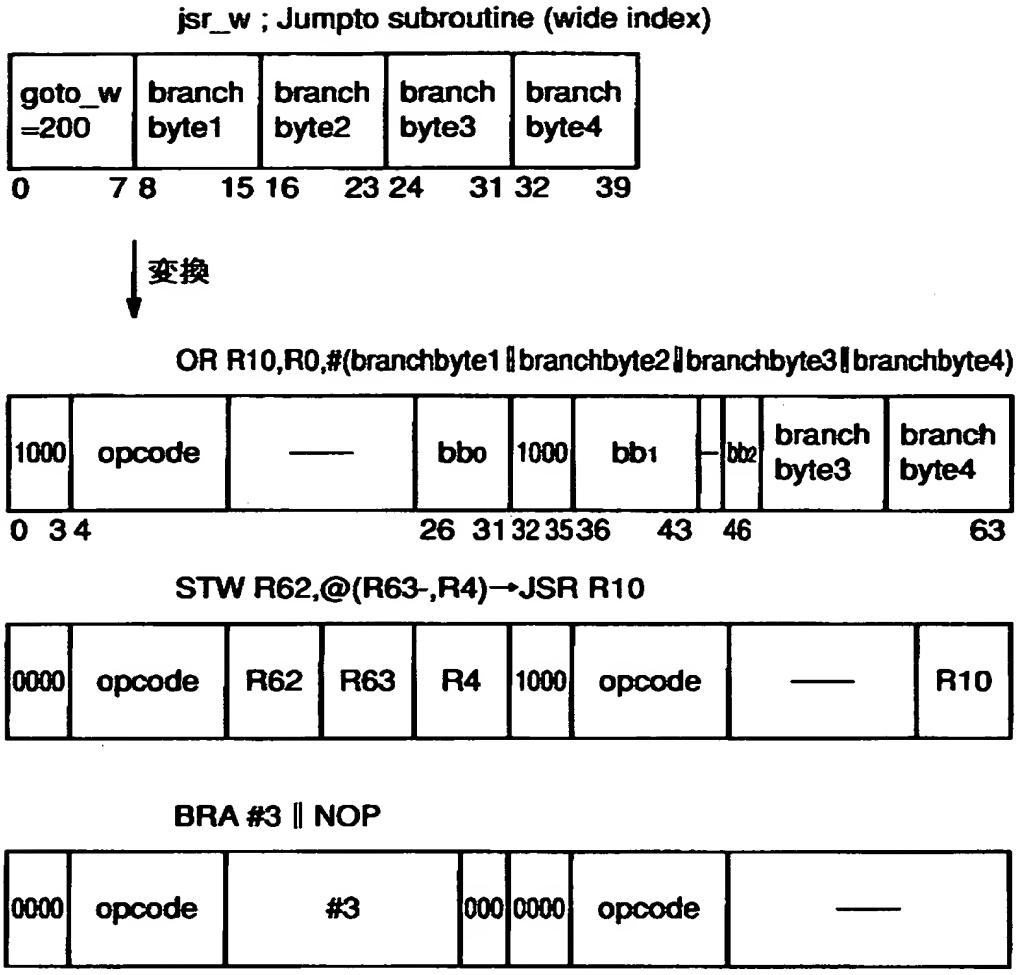
【図２７】


iload ; Load integer from local variable
iload =21
vindex
0 7 8 15
R4=-4
R10=base_addr_of_local_variable
変換
ADD R50,Rφ,#(o ‖ vindex)
1000
opcode
R50
Rφ
O
1000
O
-
O
vindex
0 3 4
26 31 32 35 36
55 56 63
STW R62,@(R63-,R4)→LDW R62,@(R10,R50)
0000
opcode
R62
R63
R4
1000
opcode
R62
R10
R50
0 3 4
31 32 35 36
63
2byte→2VLIW命令の例

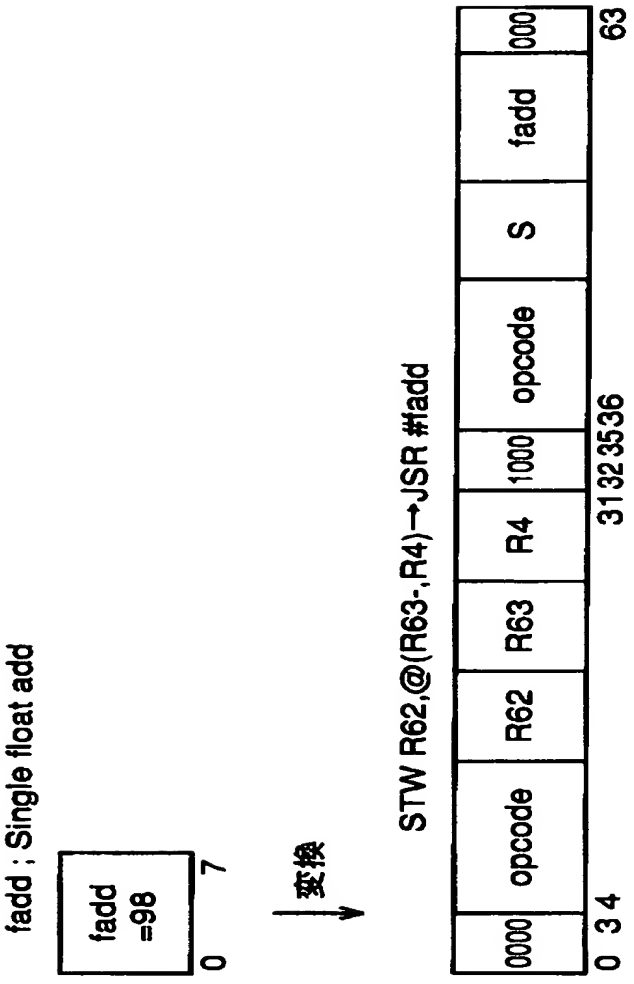
【図２８】


ifeq ; Branch if equal to 0
ifeq =153
branch byte1
branch byte2
0 7 8 15 16 23
変換
ADD R61,R62,Rφ ‖ NOP
1000
opcode
R61
R62
Rφ
0000
opcode
——
0 3 4 26 31 32 35 36 63
LDW R62,@(R63+,Rφ) ‖ NOP
0000
opcode
R62
R63
Rφ
0000
opcode
——
0 3 4 31
BRATZR R61,#(s ‖ branchbyte1 ‖ branchbyte2)
0000
opcode
R61
O
S
0000
S
–
S
branchbyte1 ‖ branchbyte2
26 31 32 35 36 43 46

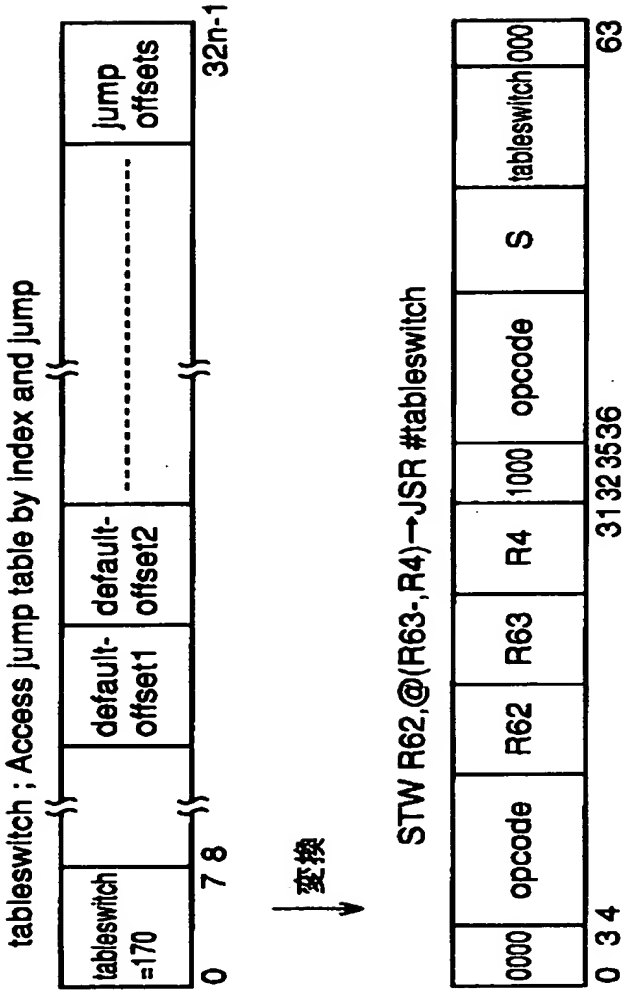
【図２９】


jsr_w ; Jumpto subroutine (wide index)
goto_w =200
branch byte1
branch byte2
branch byte3
branch byte4
0 7 8 15 16 23 24 31 32 39
変換
OR R10,R0,#(branchbyte1‖branchbyte2‖branchbyte3‖branchbyte4)
1000
opcode
——
bbo
1000
bb1
bb2
branch byte3
branch byte4
0 3 4 26 31 32 35 36 43 46 63
STW R62,@(R63-,R4)→JSR R10
0000
opcode
R62
R63
R4
1000
opcode
——
R10
BRA #3 ‖ NOP
0000
opcode
#3
000
0000
opcode
——

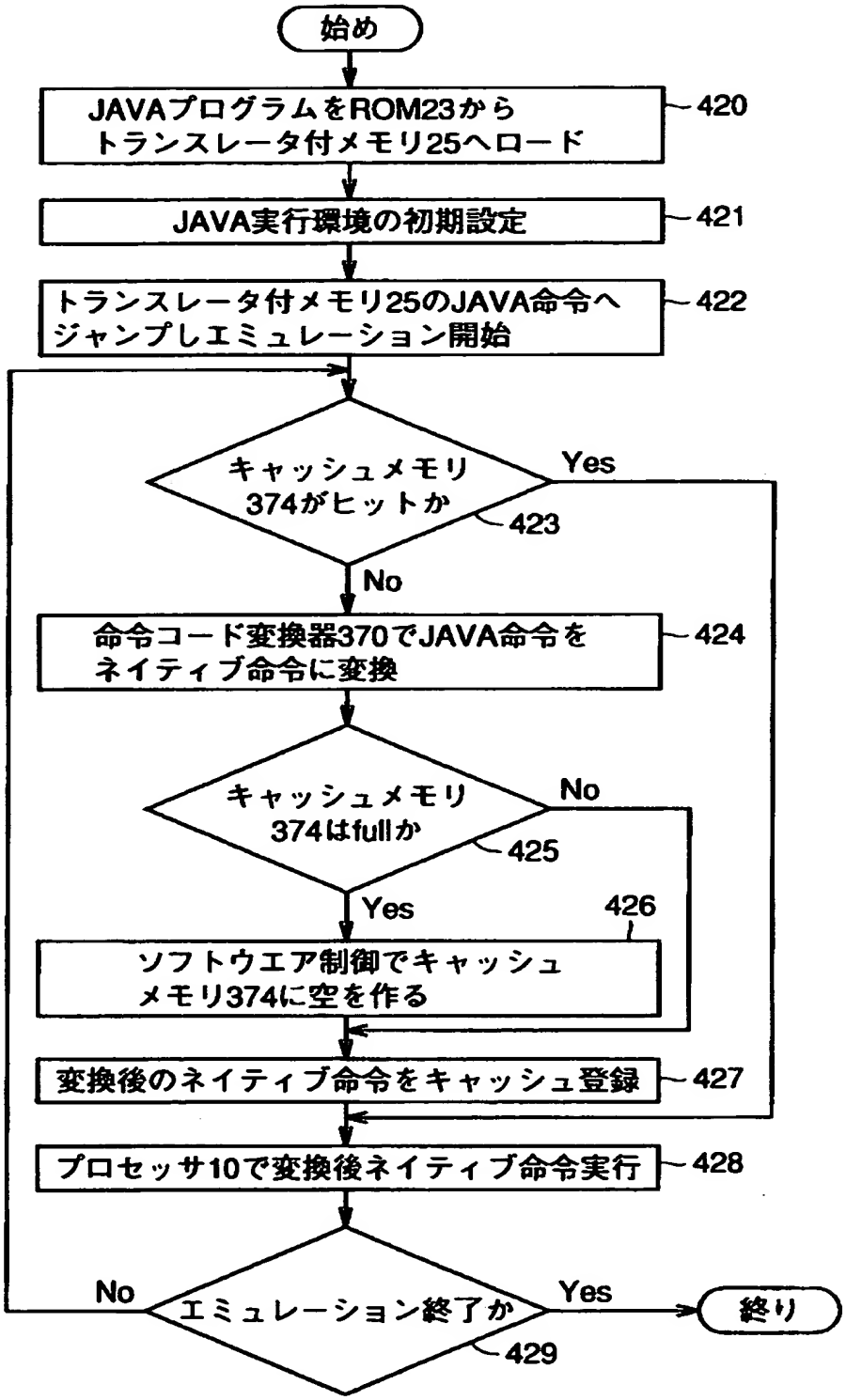
【図３０】


fadd ; Single float add
fadd
=98
0
7
変換
STW R62,@(R63-,R4)→JSR #fadd
0000
opcode
R62
R63
R4
1000
opcode
S
fadd
000
0
3
4
31
32
35
36
63

【図３１】


tableswitch ; Access jump table by index and jump
tableswitch =170
default-offset1
default-offset2
jump offsets
0
7 8
32n-1
変換
STW R62,@(R63-,R4)→JSR #tableswitch
0000
opcode
R62
R63
R4
1000
opcode
S
tableswitch
000
0 3 4
31 32 35 36
63

【図３２】


始め
JAVAプログラムをROM23から
トランスレータ付メモリ25へロード
420
JAVA実行環境の初期設定
421
トランスレータ付メモリ25のJAVA命令へ
ジャンプしエミュレーション開始
422
キャッシュメモリ
374がヒットか
423
Yes
No
命令コード変換器370でJAVA命令を
ネイティブ命令に変換
424
キャッシュメモリ
374はfullか
425
No
Yes
426
ソフトウエア制御でキャッシュ
メモリ374に空を作る
変換後のネイティブ命令をキャッシュ登録
427
プロセッサ10で変換後ネイティブ命令実行
428
エミュレーション終了か
429
No
Yes
終り

【書類名】　　要約書

【要約】

【課題】　プロセッサのハードウェア自体は変更せずに、複数個の異なる命令体系の命令からなるプログラムをネイティブ命令を用いて高速に実行可能で、大容量のメモリを必要としない命令トランスレータを提供する。

【解決手段】　命令トランスレータは、プロセッサが実行すべき命令のアドレスを受けて命令メモリから対応する命令を読出しネイティブ命令に変換するための命令コード伸長部３５０と、命令コード伸長部３５０により変換されたネイティブ命令を、命令メモリにおけるアドレスと関連付けて一時的に保持するためのキャッシュメモリ３５４と、プロセッサが実行すべき命令がキャッシュメモリ３５４に保持されているか否かの判定結果にしたがって、命令コード伸長部３５０の出力する命令と、キャッシュメモリ３５４に保持されていた対応のネイティブ命令とを選択的に出力するためのＭＵＸ３５６とを含む。

【選択図】　　図３

出 願 人 履 歴 情 報

識別番号　〔０００００６０１３〕

１．変更年月日　１９９０年　８月２４日
［変更理由］　新規登録
住　所　東京都千代田区丸の内２丁目２番３号
氏　名　三菱電機株式会社